Calhoun: The NPS Institutional Archive

DSpace Repository

Reports and Technical Reports All Technical Reports Collection

1976-05

A program for the numerical solution of large sparse systems of algebraic and implicitly defined stiff differential equations

Franke, Richard

Monterey, California. Naval Postgraduate School

http://hdl.handle.net/10945/30101

Downloaded from NPS Archive: Calhoun

Calhoun is a project of the Dudley Knox Library at NPS, furthering the precepts and goals of open government and government transparency. All information contained herein has been approved for release by the NPS Public Affairs Officer.

Dudley Knox Library / Naval Postgraduate School
411 Dyer Road / 1 University Circle
Monterey, California USA 93943

http://www.nps.edu/library

NPS-53Fe76051

NAVAL POSTGRADUATE SCHOOL

Monterey, California

A PROGRAM FOR THE NUMERICAL SOLUTION OF LARGE SPARSE SYSTEMS OF ALGEBRAIC AND IMPLICITLY DEFINED STIFF DIFFERENTIAL EQUATIONS

by

Richard Franke

May 1976

Technical Report For Period
October 1975 - April 1976

Approved for public release; distribution unlimited

FEDDOCS
D 208.14/2:
NPS-53FE76051

NAVAL POSTGRADUATE SCHOOL
Monterey, California

Rear Admiral Isham Linder
Superintendent

Jack R. Borsting
Provost

ABSTRACT

This report documents a program for the numerical solution of large sparse systems of algebraic and implicitly defined stiff differential equations. The principal use is intended to be the solution of differential equations arising from time dependent partial differential equations when the finite element method is used to discretize the space domain. The use of compact matrix storage techniques and iteration for the solution of the quasi-Newton iterates in Gear's method makes the program extremely efficient both in terms of storage requirements and execution times.

Reproduction of all or part of this report is authorized.

This report was prepared by:

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER NPS-53Fe76051	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) A Program for the Numerical Solution of Large Sparse Systems of Algebraic and Implicitly Defined Stiff Differential Equations		5. TYPE OF REPORT & PERIOD COVERED Technical Report Oct. 1975 - April 1976
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Richard Franke		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Foundation Research Program Naval Postgraduate School Monterey, California 93940		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 61152N;RR 00-01-10; N0001476WR60052
11. CONTROLLING OFFICE NAME AND ADDRESS Chief of Naval Research Arlington, Virginia 22217		12. REPORT DATE May 1976
		13. NUMBER OF PAGES
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Gear's Method
Differential-Algebraic systems
Stiff equations
Sparse matrices
Finite element method

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This report documents a program for the numerical solution of large sparse systems of algebraic and implicitly defined stiff differential equations. The principal use is intended to be the solution of differential equations arising from time dependent partial differential equations when the finite element method is used to discretize the space domain. The use of compact matrix storage techniques and iteration for the solution of the quasi-Newton iterates in Gear's method makes the program extremely efficient both in terms

of storage requirements and execution times.

1.0 Introduction

This report documents a program for the solution of algebraic and implicitly defined stiff differential equations. We were particularly interested in solving very large systems of differential equations arising from partial differential equations where the finite element method has been applied in the space variables.

Our original goal was to use a compact storage scheme for the large matrices involved and to use iteration to solve the linear algebraic systems which occur. However, the resulting program is easily adapted to different applications through user modifications accomplished by replacement of one or two relatively simple subroutines. Thus the program is a powerful one which can be used in a variety of applications. Four examples, illustrating different matrix storage techniques and different linear equation solvers are given in the appendix. Other storage schemes and solution methods, e.g. symmetric Jacobian with Cholesky decomposition, are relatively simple to implement.

In Section 2 a brief review of the integration scheme is given. In Section 3 a discussion of differences between this program and the one from which it was adapted is given. Section 4 is devoted to a discussion of information concerning the use of this package.

2.0 Theoretical Background

Consider the system of $N = m + \ell$ differential and algebraic equations in $y_1,\ldots,y_m,v_1,\ldots,v_\ell$,

$$F(y,\dot{y},t) + P(t)\,V = 0\ , \tag{1}$$

with all or some of the initial values $y_1(t_0),\ldots,y_m(t_0),v_1(t_0),\ldots,$ $v_\ell(t_0)$ specified. Enough of the above values must be given in order to determine the remaining values and initial values for any of the derivatives, $\dot{y}_1,\ldots,\dot{y}_m$ which appears in equation (1). In equation (1), $P(t)$ is an $N \times \ell$ matrix, F is a vector with N components, and V is a vector.

The program documented here is a modification of a program due to Brown and Gear [1]. The method of solution is a modification of Gear's method for stiff differential equations [2]. The application to differential algebraic systems was given by Gear [3]. We will briefly describe the method here for completeness, and refer the reader to the references for more details.

Suppose that approximate solution values are known at a number of equally spaced points, $t_{n-1},t_{n-2},\ldots,t_{n-k}$, and are represented by $y^{(n-1)},\ldots,y^{(n-k)}$ respectively. Let $V(t_{n-1})$ be represented by $V^{(n-1)}$. Use of a backward differentiation formula gives

$$h\,\dot{y}^{(n)} = \frac{1}{\beta_0}\left(\alpha_0\, y^{(n)} + \alpha_1\, y^{(n-1)} + \ldots + \alpha_k\, y^{(n-k)}\right),$$

where $h = t_{i+1} - t_i$. The coefficients α_i and β_0 are from Gear [2], p. 217. Substitution of this into (1) gives

$$\text{(2)} \qquad F\left(y^{(n)}, -\frac{\alpha_0}{\beta_0 h}\, y^{(n)} + \textstyle\sum_n,\ t_n\right) + P(t_n)\, V^{(n)} = 0$$

as the equation which $y^{(n)}$ and $V^{(n)}$ must satisfy. In equation (2),

$$\textstyle\sum_n = \dfrac{1}{\beta_0 h}\left(\alpha_1\, y^{(n-1)} + \ldots + \alpha_k\, y^{(n-k)}\right).$$

In general, equation (2) represents a system of nonlinear equations for $y^{(n)}$ and $V^{(n)}$. The method used for solving this system of algebraic equations is a variant of Newton's method. The initial guess, $y^{(n),0}$, is obtained by polynomial extrapolation using a Hermite polynomial interpolating the known values $\dot{y}^{(n-1)}, y^{(n-1)}, \ldots, y^{(n-k)}$. Thus the predicted values has the form

$$(3)\quad y^{(n),0} = h\,\bar{\beta}_1\,\dot{y}^{(n-1)} + \bar{\alpha}_1\,y^{(n-1)} + \ldots + \bar{\alpha}_{n-k}\,y^{(n-k)} .$$

The application of Newton's Method to equation (2) then yields the corrector equation

$$(4)\quad J \cdot \begin{pmatrix} y^{(n),i+1} - y^{(n),i} \\ V^{(n),i+1} - V^{(n),i} \end{pmatrix} = -F\left(y^{(n),i}, -\frac{\alpha_0}{\beta_0 h}\,y^{(n),i}, t_n\right) + P(t_n)\,V^{(n),i} ,$$

where J is the Jacobian matrix,

$$(5)\qquad J = \left(\frac{\partial F}{\partial y} - \frac{\alpha_0}{\beta_0 h}\frac{\partial F}{\partial \dot{y}} \;\middle|\; P \right) .$$

Gear shows that the initial guess for $V^{(n)}$ is not important, and thus $V^{(n-1)}$ is used. Up to three iterations are performed on the corrector equation. The matrix J is not evaluated at each iteration, nor even at each timestep. J is evaluated whenever (i) the timestep or order is changed, or (ii) the corrector iteration fails to converge in three iterations.

If the corrector iteration fails to converge, the J matrix is evaluated, unless it had already been evaluated at the current time.

If it has been evaluated at the current time, the timestep is reduced by a factor of 4. In either case the step is then retired.

If the corrector iteration converges, the local error is estimated, based on the fact that the local error is approximately proportional to the difference between the predicted and corrected values of $y^{(n)}$. For this purpose a relative error tolerance is used for large solutions and an absolute error tolerance for small solutions. The root-mean-square norm (Euclidean norm divided by the square root of the number of components) is used for the vector with components $e_i/ymax_i$, where e_i is the estimated local error in $y_i^{(n)}$ and $ymax_i = \max\limits_{0 \le k \le n} (|y_i^{(k)}|, 1)$. If the error is larger than that specified by the user, an acceptable timestep is estimated for the current order or order one lower, and the step repeated. Up to three such failures are permitted, after which an attempt is made to start over with a first order method.

When using a method of order q, the program takes at least $q + 1$ steps before changing the timestep. Changes in timestep are preceeded by calculation of the predicted timestep at current order and order one higher and one lower. If the timestep can be increased by more than 10%, the order corresponding to the largest possible timestep is used. If the timestep cannot be increased by at least 10%, the current order and stepsize are retained for at least 10 more steps.

After each step a test is made to determine whether time has advanced to or beyond the input end time. Control is returned to the calling program if it has.

At the initial call, no history of the solution is available, so the program must begin with a first order method, taking two such steps in accordance with the above description. The timestep must be suitably small, again in accordance with the above. At the point the program can begin to increase the order of the method and the timestep. Because the Jacobian matrix must be generated whenever the timestep is changed, it is not efficient to try too large a timestep initially. Because the program rapidly finds the best order and timestep, it is relatively cheap to underestimate the initial timestep compared to the cost of overestimating it.

3.0 Differences compared with DFASUB

The principal differences between the SDESOL/LDASUB package and DFASUB, and the reasons for incorporating them are as follows.

(i) The main goal of this revision was to generate a program which could solve very large sparse systems of differential equations efficiently, both in terms of storage requirements and execution time. We are particularly interested in the solution of ordinary differential equations arising from partial differential equations where the finite element method has been used to discretize the problem in space.

Large sparse problems require at least a different system of storing the Jacobian matrix and possibly the use of iteration to solve for the quasi-Newton iterates in equation (2.4). Two such subroutine packages, to be used with the basic subroutines, have been provided. Another package using standard elimination techniques is also provided and is convenient for smaller systems of equations. Use of any of these options requires the user to supply a subroutine to evaluate his form of the equations (1), and for efficiency, a subroutine to explicitly evaluate the Jacobian. A subroutine to approximate a full Jacobian through numerical differencing has been provided. With the exception of a minor correction, this is the same as given in [1]. While use of this routine is convenient, it is inefficient and should be avoided for large systems. It is anticipated that the user can provide his own subroutine package, using his own storage scheme for the Jacobian, and with a suitable equation solver for the Newton iterates. There should be no need to disturb the basic package which carries out the time integration.

(ii) For user convenience, without a major rewrite of DFASUB, a driver routine, SDESOL, to be referenced by the user and which then communicates with LDASUB was written. The chief function of SDESOL is to set up a number of references to work storage areas required by LDASUB. In addition, some testing of parameters is accomplished, and a subroutine to calculate initial values of derivatives is called.

(iii) A subroutine to calculate initial values of derivatives, DERVAL, has been provided. The routine provided requires that the first m rows of $\frac{\partial F}{\partial \dot{y}}$ be nonsingular, which does not need to be true in the general case. For this reason, and others discussed in Section 4, the user may need to provide either his own version of DERVAL to evaluate the derivatives initially, or else he may supply initial values and a dummy version of DERVAL.

(iv) Other changes made in generating LDASUB from DFASUB were to simplify the code for the particular type of problem we wish to solve, while others were to enhance the usability of the code. Some errors were also corrected, notably two errors in coefficients for the fifth and sixth order methods. DFASUB had the capability of computing various elements of the Jacobian at different times if they had different dependencies, with the possibility of inverting that part of the matrix at that time, if it could be done. This could result in increased efficiency in certain problems, at some expense in convenience, but for our purposes it was not considered useful, and was removed. Therefore only one call is made to evaluate the Jacobian. The Jacobian was evaluated at the beginning of each timestep in DFASUB, but this has been eliminated in LDASUB, in accordance with the description in Section 2. A subroutine, S2, was called from DFASUB to evaluate time dependent terms whenever time was changed. This is reasonable, since the function evaluation routine may be called several times at a given value of the independent variable. We have removed this, preferring to test for a new time in the routines where time dependent terms appear, then evaluating and storing them internally to that routine when necessary. This helps make the function evaluation more self contained, as well.

In DFASUB extra parameters in the calling sequence allowed the user to communicate constants to the function evaluation and Jacobian subroutines via DFASUB. We believe this is inefficient and confusing, and removed this capability, preferring to communicate from the main program to these subroutines via Common, or possibly through multiple entry points.

The norm used for error tests in DFASUB is the Euclidean norm. This has the undesirable property that for large systems the allowable error criterion may be large. We therefore changed to the root-mean-square norm in LDASUB, which is simply the Euclidean norm divided by the square root of the number of components. One other change was made in the error tests. As noted in Section 2, the error vector has components $e_i/ymax_i$, where e_i is the estimated local error in the i^{th} variable y_i , and $ymax_i = \max_{0 \le k \le n} (|y_i^{(k)}|, 1)$. In DFASUB the maximum was taken only up to the previous timestep, $n-1$. This change was incorporated because some of the problems in which we were interested began with many components at zero, but which very rapidly became large, around 10^{12} or more. Without updating the value of $ymax_i$, the size of the timestep was artificially kept extremely small in order to satisfy an unreasonable error tolerance. For this reason, the maximum value of the component was updated before the norm of the relative errors was computed. For problems where values range near to one, the modification will result in no appreciable change in performance.

The printout of counters, timestep, time, and values of the dependent variables was made an option through a parameter in the calling sequence. An additional value printed is the order of the method being used.

DFASUB incorporated the capability of terminating if a certain number of floating point overflows had occurred. This capability was removed from LDASUB.

The final modification to the program was the incorporation of a restart capability without having to begin again with a first order method. This was accomplished by adding two entry points to LDASUB. One, LDASAV, saves values internal to LDASUB, returning them to the main program, where they can be saved for the time at which the calculation is to be resumed. At that time, another entry point, LDARST, restores those values internal to LDASUB, while other necessary values are restored through the calling sequence.

4.0 Subroutine Descriptions

The description of subroutines is divided into two subsections. The first deals with the basic integration routine and other subroutines which make up the core package, and which should not be modified by the casual user. The second deals with a set of supporting subroutines, at least one of which must be supplied by the user since it defines his system of equations. The others may be usable in the form given in one of the examples, or can be defined by the user to accomplish his desired implementation.

4.1 Basic Subroutine Package

4.1.1 Subroutine SDESOL

This routine is the only one which needs to be referenced by most users. It serves as a driver for the integration routine, LDASUB. SDESOL has a simpler calling sequence than LDASUB and relieves the user of having to set up a number of auxiliary storage arrays. In addition, the routine calls DERVAL to calculate the values of the derivatives on the initial call.

The calling sequence is

```
CALL  SDESOL(Y,YL,T,TEND,NY,NL,M,JSKF,MAXDER,IPRT,H,HMIN,HMAX,RMSEPS,W)
```

where the parameters are defined as follows.

Y - Input and output. An array dimensioned (7,NY). On the initial call this array contains the initial values of the dependent variables y_i, i=1, ..., m in Y(1,i) . During execution of the program the approximate values of $\frac{d^j y_i}{dt^j} \cdot \frac{h^j}{j!}$ is stored in Y(j+1,i) . Here h is the current stepsize. These values must not be changed between returns to the calling program and subsequent entries to SDESOL. It is possible to interpolate for values of the dependent variables at times other than those calculated by using the formula $y_i(t+s) = \sum_{j=0}^{q} Y(j+1,i)\left(\frac{s}{h}\right)^j$, where q is the order formula currently in use, and is obtained as $q = |JSKF/10|$.

YL - input and output. Array of linear variables, v_i, i=1, ..., ℓ . The user supplies initial values for these variables, and during execution it contains current values of the linear variables.

T - input and output. The user supplies the initial time, which is updated to current time during execution.

TEND -input. Time at which the integration is to end. This is the only parameter normally changed by the user between successive entries to SDESOL.

NY - input. The number of differential and nonlinear variables, m .

NL - input. The number of linear variables, ℓ . This may be zero.

M - input. The number of variables to be included in the local error test. The error test will be performed for the variables y_i, i=1, ..., M . The M used is no greater than NY .

JSKF -input and output. An indicator: on input,

JSKF = 0 indicates that this is the initial call to SDESOL. Initial values of the derivatives are calculated and auxiliary storage references are set up. This also indicates to subroutine LDASUB to initialize parameters and begin with a first order integration method.

JSKF > 0 indicates a continuation of a previous call to SDESOL .

JSKF = - 1 indicates a restart call. This is discussed further in Section 4.1.2.

JSKF < - 1 may result from the user neglecting to test for error returns from SDESOL. Because of this possibility, the run is terminated with an appropriate comment when JSKF < - 1 is input.

On output, JSKF normally is a two digit number, ± qp . q is the order of the formula currently being used for the integration. p is an indicator determining the type of return. JSKF > 0 , p = 1 is the normal return. Note that SDESOL may be re-entered to continue the solution without changing JSKF. JSKF < 0 is an error return, with the value of p indicating the error, as follows.

p = 1 error test failure for H ≥ HMIN

p = 3 corrector failed to converge for H > HMIN

p = 4 corrector failed to converge for a first order method

p = 5 error return from subroutine NUITSL

p = 6 error return from subroutine DERVAL

MAXDER-input. Maximum order method which should be used. The highest order possible is six.

IPRT- input. Print control indicator.

≤ 0 , no print from LDASUB

> 0 , at each step, print number of steps, number of Jacobian evaluations, current order being used, stepsize for next step, current time, and current values of the dependent variables.

H - input and output. On initial call it is an estimate of the timestep. During execution it is updated to the current value, and on return contains the stepsize to be tried for the next step. The input value need not be accurate. It is better to underestimate than to overestimate the initial value. The stepsize and order are varied to meet the local error tolerance specified. The user does not normally change the stepsize between entries to SDESOL.

HMIN- input. The minimum stepsize to be allowed.

HMAX- input. The maximum stepsize to be allowed.

RMSEPS-input. The error test constant. The values of the relative local errors must have root-mean-square norm less than RMSEPS.

W - an array of auxiliary storage required by LDASUB. This array must contain a total number of locations equal to the sum of (i) 13*NY + 5*NL for arrays used in LDASUB, (ii) storage for the Jacobian matrix, and (iii) any locations used in processing the Jacobian, e.g., scratch storage used by an equation solver.

4.1.2 Subroutine LDASUB

This subroutine is the basic integration routine and performs the process in essentially the same manner as subroutine DFASUB. A brief description is given in Section 2 and differences between this routine and DFASUB are outlined in Section 3. Parameters in this routine in which the user may be interested are stored in the W array, an argument of subroutine SDESOL.

YMAX - array of maximum magnitudes of the independent variables, y_i, up to the current time (or one, if less than one). This is stored beginning at location 7*NY + NL + 1 of the W array.

ER - This is the array of differences between the predicted and corrected values of the variables, y_i, and is proportional to the estimated error. This array is stored beginning in location 8*NY + NL + 1 of the W array.

This subroutine incorporates a restart capability. In order to restart from a previous point without beginning again with a first order method, it is necessary to have saved a number of variables internal to LDASUB, and then restore them before calling SDESOL again. To save the internal parameters, the user calls subroutine LDASAV(SAV). Here SAV is an array of length 29 in which the values to be saved will be stored. In addition to SAV, the user must also save a number of arrays in the calling sequence of LDASUB, and this is most easily accomplished by saving the W array in the calling sequence for SDESOL. Once these arrays have been saved, along with the other simple parameters in the calling sequence (Y and YL need not be saved), the user is free to use the package to solve a different problem, or to terminate the computer run, to be restarted later.

At the time the problem is to be restarted, the user calls subroutine LDARST(SAV), where SAV is the array of values obtained previously by calling LDASAV. This restores internal values in LDASUB. The user then calls SDESOL with the same simple parameters and the W array as before, except that JSKF = - 1 and a new end time, TEND, is provided. Restoration of values (including Y and YL) in LDASUB is completed and solution of the problem resumes.

If the user desires to change the error tolerance, number of variables in the error test, or maximum order to be used, the user must make a new initial call to SDESOL, that is, set JSKF = 0 .

4.1.3 Subroutine COPYZ

This subroutine simply transfers the contents of one array into another array.

4.2 Supporting Subroutines

This group of subroutinesmust, at least in part, be supplied by the user. The user must supply at least one subroutine, DIFFUN. For better efficiency, the user should supply a subroutine, JACMAT, to explicitly evaluate the Jacobian, although a version which approximates the Jacobian by numerical differencing is given in the appendix. To take advantage of sparsity or other features of his problem, the user will need to supply the subroutine NUITSL to solve the systems of equations (2.4). For certain problems the user may have to supply subroutine DERVAL to calculate the initial values of the derivatives. We discuss the requirements of these subroutines in turn.

4.2.1 Subroutine DIFFUN

This subroutine simply evaluates the equations (2.1) at a given time and values of y , $\dot{y}$, and V . Other parameters in the function definition must be transmitted from the calling program via COMMON or some other device, determined by the user.

The calling sequence is

CALL DIFFUN (Y, YL, T, HINV, DY), where the parameters are defined as follows.

- Y - input. Same as in SDESOL. This array contains the current values of the variables y_i and their (scaled) derivatives.
- YL - input. Same as in SDESOL. This array contains the current values of the linear variables.
- T - input. Current time.
- HINV - input. $1/h$, where h is the current stepsize.
- DY - output. Array of function values.

4.2.2 Subroutine JACMAT

This subroutine evaluates the Jacobian matrix J , equation (2.5) at the given time and current values of the dependent variables, order, and stepsize. A version of JACMAT which approximates J by numerical differencing is given in the appendix. For maximum efficiency, the user should supply the explicit representation of the Jacobian. Because the Jacobian is used to solve for the quasi-Newton iterates, it is not necessary for the Jacobian to be exact. Thus the user should consider the possibility of approximations which reduce the total number of computations in this step, with due regard for the fact that a smaller timestep may be required to obtain convergence of the corrector within three iterations.

The calling sequence for this subroutine is

CALL JACMAT (Y, YL, T, HINV, A2, N, NY, EPS, DY, F1, PW), where the parameters are defined as follows.

- Y - input. Same as in SDESOL, Y contains the current values of the variables y_i and their (scaled) derivatives.
- YL - input. Same as in SDESOL. This array contains the current values of the linear variables.
- T - input. Current time.
- HINV - input. $1/h$, where h is the current stepsize.
- A2 - input. The constant α_0/β_0 from LDASUB.
- N - input. Total number of variables.
- NY - input. Number of differential and nonlinear variables.
- EPS - input. Error constant from LDASUB, $\sqrt{M}\cdot$RMSEPS.
- DY - input. Array of current function values.
- F1 - scratch array of N locations available for use by this routine.
- PW - output. The Jacobian matrix J, or an approximation, calculated in JACMAT and returned to calling program. This matrix is used in subroutine NUITSL and the storage mode must agree between the two subroutines.

4.2.3 Subroutine NUITSL

This subroutine solves the equations (2.4) for the quasi-Newton iterates. This subroutine will normally be supplied by the user, although versions which solve the system by elimination methods and iterative methods, respectively, are given in the examples in the appendix. This subroutine will often be modified or replaced by the user to take advantage of sparsity or other features of his problem in connection with JACMAT, of course.

The calling sequence for this subroutine is

CALL NUITSL (PW, DY, F1, N, NY. EPS, YMAX, NEWPW, KRET), where the parameters are defined as follows.

PW - input. The Jacobian matrix J computed in JACMAT.

DY - input. Right hand side of the linear system to be solved.

F1 - output. The solution is returned in the array F1 .

N - input. Total number of variables.

NY - input. Number of differential and nonlinear variables.

EPS - input. Error constant from LDASUB, $\sqrt{M} \cdot$ RMSEPS .

YMAX - input. Array of maximum magnitudes of y_i up to the current time (or one if maximum magnitude is less than one).

NEWPW - input. Indicates whether a new J matrix has been computed since the last entry to NUITSL.

= 1, indicates this is a new J matrix. If any preprocessing, such as LU decomposition is to be done, the preprocessing should be done and NEWPW set to zero.

= 0, indicates the J matrix is the same as on the previous entry to NUITSL.

KRET - output. Return indicator.

= 0 , normal return

= 1 , error return, solution of equations not obtained.

Note that the parameters EPS and YMAX are useful if an iterative method is used for solutuon of the equations. Because the solution represents corrections to the predicted value, and corrections to that, the solution is small compared to the dependent variable values. Hence, compared to the YMAX array, the error tolerance can be fairly large. The following convergence criteria have been used, with great success. Let δu_i denote the $i^{\underline{th}}$ component of the difference between successive iterates, with u_i being the $i^{\underline{th}}$ component of the current iterate. Then the iteration is considered to have converged whenever

$$\text{(i)} \qquad \sum_{i=1}^{NY} \left(\frac{\delta u_i}{Ymax_i} \right)^2 < \left(\varepsilon/100 \right)^2 \text{, or}$$

$$\text{(ii)} \qquad \sum_{i=1}^{NY} \left(\frac{\delta u_i}{\max(|u_i|, \varepsilon)} \right)^2 < \varepsilon^2 \text{.}$$

Condition (i) requires convergence to 2 digits more accuracy than the user has asked for in the solution of the system (2.1), relative to YMAX. Condition (ii) requires the same relative accuracy in u_i as is asked for by the user in the solution of the system (2.1), unless the solution is smaller than ε , in which case the change is compared to ε rather than $|u_i|$. This avoids difficulty if u_i is close to zero. The ε above is EPS = $\sqrt{M}\cdot$RMSEPS, where M and RMSEPS are inputs to SDESOL. Two versions of NUITSL incorporating iteration and this convergence test are given in the appendix.

4.2.4 Subroutine DERVAL

This subroutine solves for, or otherwise supplies the initial values of the derivatives, and possibly other variables. In some instances it may need to be supplied by the user. The standard version of DERVAL given in the appendix uses Newton's method to solve the first m (=NY) of the equations (2.1) for $\dot{y}(t_o)$, assuming values for $y(t_o)$ and $V(t_0)$ have been supplied. To accomplish this, the matrix $\frac{\partial F}{\partial \dot{y}}$ is needed, and this is obtained by calling JACMAT with $h = 16^{-20}$, $A2 = -1$, and $N = NY$, implying $NL = 0$ for this call. Special care must be taken if in fact NL is not zero to assure that the matrix is computed and stored properly. The matrix returned is then $16^{20} \frac{\partial F}{\partial \dot{y}}$. A call to DIFFUN yields the function values $F(y(t_0), \dot{y}(t_0), t_0) + PV(t_0)$ where $\dot{y}(t_0)$ is the current iterate. Multiplication of the function values by 16^{20} and a call to NUITSL (again with $N = NY$) gives the Newton iterate. Of course, the same sort of special care as necessary in JACMAT is necessary in NUITSL.

Obviously the above scheme cannot work if $\frac{\partial F}{\partial \dot{y}}$ is singular, such as it would be if one of the equations is algebraic. In this instance the user must either devise his own version of DERVAL, or supply the values along with a dummy version of DERVAL. In an extreme case the user may simply set initial derivatives to zero. This will provide a poor predicted value on the first step, and will force an artificially small timestep for the first two steps. However, the overall penalty is generally small, as appropriate (corrected) values are computed at the first step, and after two steps the program quickly increases the timestep.

The calling sequence for this subroutine is

CALL DERVAL (Y, YL, T, N, NY, DY, KERET) , where the parameters are defined as follows.

Y - input and output. Same as in SDESOL. On entry Y(1,i) contains the initial values of the variables y_i . On return, the values of the derivatives are stored in Y(2 i) .

YL - input. Same as in SDESOL. This array contains the initial values of the linear variables.

T - input. initial time

N - input. Total number of variables.

NY - input. Number of differential and nonlinear variables.

W - The scratch array from SDESOL, can be used in any way needed by this subroutine.

KERET - output. Return indicater

= 0 normal return

= 1 error return, initial values were not obtained.

5.0 Acknowledgement

The author wishes to express his thanks to Professors David Salinas and Dong Nguyen of the Mechanical Engineering Department at the Naval Postgraduate School. They supplied the initial applications and encouragement for this work. They have continued to support it through valuable discussions with the author throughout the development period.

Appendix 1: Program Listings

The following are listings of the basic subroutine package and supporting subroutines which are of general use. For simple problems the user only needs to supply a calling program and a subroutine, DIFFUN, to evaluate the equations. Use of the NUITSL routine in computer facilities which do not subscribe to the IMSL package will necessitate modifications to replace LUDATF with another LU decomposition routine, and LUELMF with another forward and backward substitution routine.

```
      SUBROUTINE SDESOL (Y,YL,T,TEND,NY,NL,M,JSKF,MAXDER,IPRT,H,HMIN,
     1HMAX,RMSEPS,W)
C
C     ------------------------------------------------------------------
C
C     SUBROUTINE SDESOL IS A DRIVER ROUTINE FOR SUBROUTINE LDASUB.
C     ITS PURPOSE IS TO SET UP THE NECESSARY REFERENCES TO A LARGE
C     BLOCK OF AUXILLARY STORAGE, AND OBTAIN INITIAL VALUES OF
C     DERIVATIVES.
C     THE CALLING SEQUENCE FOR SDESOL IS
C
C CALL SDESOL(Y,YL,T,TEND,NY,NL,M,JSKF,MAXDER,IPRT,H,HMIN,HMAX,RMSEPS,W)
C
C     WHERE THE PARAMETERS ARE DEFINED AS FOLLOWS.
C
C        Y        - ARRAY DIMENSIONED (7,NY).  THIS ARRAY CONTAINS THE
C                   DEPENDENT VARIABLES AND THEIR SCALED DERIVATIVES.
C                   Y(J+1,I) CONTAINS THE J-TH DERIVATIVE OF THE I-TH VAR
C                   IABLE TIMES H**J/J-FACTORIAL, WHERE H IS THE CURRENT
C                   STEPSIZE.  ON FIRST ENTRY THE CALLER SUPPLIES THE
C                   INITIAL VALUES OF EACH VARIABLE IN Y(1,I).  ON SUB-
C                   SEQUENT ENTRIES IT IS ASSUMED THE ARRAY HAS NOT
C                   BEEN CHANGED.  TO INTERPOLATE TO NON-MESH POINTS,
C                   THESE VALUES CAN BE USED AS FOLLOWS.  IF H IS THE
C                   CURRENT STEPSIZE AND VALUES AT TIME  T+E  ARE
C                   NEEDED, LET  S = E/H  AND THEN
C
C                                                JS
C                         I-TH VARIABLE AT  T+E IS  SUM Y(J+1,I)*S**J
C                                                J=0
C
C                   THE VALUE OF JS IS OBTAINED IN THE CALLING PROGRAM
C                   BY   JS = IABS(JSKF/10)
C        YL       - ARRAY OF NL VARIABLES WHICH APPEAR LINEARLY.
C        T        - CURRENT VALUE OF THE INDEPENDENT VARIABLE (TIME)
C        TEND     - END TIME
C        NY       - NUMBER OF DIFFERENTIAL EQUATIONS AND NONLINEAR
C                   VARIABLES.
C        NL       - NUMBER OF LINEAR VARIABLES
C        M        - NUMBER OF VARIABLES INCLUDED IN THE ERROR TEST
C        JSKF     - AN INDICATOR USED BOTH ON INPUT AND OUTPUT
C                   ON INPUT, JSKF = -1 INDICATES A RESTART CALL TO
C                   SDESOL.  JSKF = 0 INDICATES AN INITIAL CALL TO
C                   SDESOL.  JSKF > 0 INDICATES A CONTINUATION OF THE
C                   PREVIOUS CALL TO SDESOL.  JSKF < -1 MAY HAVE RESULTED
C                   FROM THE USER NEGLECTING TO TEST FOR ERROR RETURNS
C                   FROM SDESOL.  BECAUSE OF THIS POSSIBILITY, JSKF < -1
C                   RESULTS IN TERMINATION OF THE RUN WITH THE
C                   APPROPRIATE COMMENT.
C                   ON OUTPUT, JSKF CONSISTS OF TWO DIGITS AND SIGN,
C                   + OR - QP.  Q IS THE ORDER OF THE FORMULA CURRENTLY
C                   BEING USED.  P INDICATES THE TYPE OF RETURN, AS
C                   FOLLOWS.
C                   JSKF > 0, P = 1 IS THE NORMAL RETURN
C                   JSKF < 0 IS AN ERROR RETURN, WITH THE FOLLOWING
C                   MEANINGS.
C                         P = 1   ERROR TEST FAILURE FOR H > HMIN
C                         P = 3   CORRECTOR FAILED TO CONVERGE FOR H > HMIN
C                         P = 4   CORRECTOR FAILED TO CONVERGE FOR FIRST
C                                 ORDER METHOD
C                         P = 5   ERROR RETURN FROM SUBROUTINE NUITSL
C                         P = 6   ERROR RETURN FROM SUBROUTINE DERVAL
C        MAXDER   - MAXIMUM ORDER DERIVATIVE THAT SHOULD BE USED IN
C                   METHOD.  IT MUST BE NO GREATER THAN SIX.
C        IPRT     - INTERNAL PRINT CONTROL INDICATOR FOR LDASUB.
C                         IPRT = 0   NO PRINT
C                         IPRT > 0   PRINT COUNTERS, STEPSIZE, CURRENT TIME
C                                    AND VALUES OF DEPENDENT VARIABLES AT
C                                    EACH STEP.
C        H        - CURRENT STEPSIZE.  AN INITIAL VALUE MUST BE SUPPLIED
C                   BUT NEED NOT BE THE ONE WHICH MUST BE USED, SINCE THE
C                   SUBROUTINE WILL CHOSE A SMALLER ONE IF NECCESSARY TO
C                   KEEP THE ERROR PER STEP SMALLER THAN THE SPECIFIED
C                   VALUE.  IT IS BETTER TO UNDERESTIMATE THE INITIAL
C                   STEPSIZE THAN TO OVERESTIMATE IT.  THE STEPSIZE IS
```

```
C                   NORMALLY NOT CHANGED BY THE USER.
C        HMIN     - MINIMUM STEPSIZE ALLOWED
C        HMAX     - MAXIMUM STEPSIZE ALLOWED
C        RMSEPS   - THE ERROR TEST CONSTANT.  THE ROOT-MEAN-SQUARE OF
C                   THE SINGLE STEP ERROR ESTIMATES, ER(I), DIVIDED BY
C                   YMAX(I) = (MAXIMUM TO CURRENT TIME OF Y(I) ) MUST BE
C                   LESS THAN EPS.  THE STEPSIZE AND/OR THE ORDER
C                   ARE VARIED TO ACHIEVE THIS.
C        W        - SCRATCH STORAGE ARRAY.  MUST BE AT LEAST 13*NY + 5*NL
C                   LOCATIONS, PLUS THOSE REQUIRED FOR STORAGE OF THE
C                   MATRIX  PW (SEE DESCRIPTION OF SUBROUTINE JACMAT).
C                   THE STORAGE OF PW WILL NORMALLY REQUIRE NO MORE THAN
C                   N**2 + 2*N LOCATIONS, AND IF COMPACT STORAGE TECH-
C                   NIQUES ARE USED, CAN BE MUCH FEWER.
C
C-----------------------------------------------------------------------
      DIMENSION Y(7,1), YL(1), W(1)
      IF (JSKF.GT.0) GO TO 120
      IF (JSKF.LT.-1) GO TO 140
      N = NY+NL
      IF (JSKF.LT.0) GO TO 110
C
C     IF THIS IS THE FIRST ENTRY, OBTAIN VALUES OF THE DERIVATIVES.
      CALL DERVAL (Y,YL,T,N,NY,W,KRETR)
      IF (KRETR.NE.0) GO TO 130
C
C     NOW SET UP STORAGE BLOCKS IN THE W ARRAY.  THIS NEEDS TO BE DONE
C     ONLY INITIALLY AND ON RESTARTS.
C
C        THE ARRAY    SAVE    STARTS AT LOCATION  1      IN THE W ARRAY
C        THE ARRAY    YLSV    STARTS AT LOCATION  NSVL   IN THE W ARRAY
C        THE ARRAY    YMAX    STARTS AT LOCATION  NYMAX  IN THE W ARRAY
C        THE ARRAY    ER      STARTS AT LOCATION  NER    IN THE W ARRAY
C        THE ARRAY    ESV     STARTS AT LOCATION  NESV   IN THE W ARRAY
C        THE ARRAY    F1      STARTS AT LOCATION  NF1    IN THE W ARRAY
C        THE ARRAY    DY      STARTS AT LOCATION  NDY    IN THE W ARRAY
C        THE MATRIX   PW      STARTS AT LOCATION  NPW    IN THE W ARRAY
C
  110 NSVL = 7*NY+1
      NYMAX = NSVL+NL
      NER = NYMAX+NY
      NESV = NER+NY
      NF1 = NESV+NY
      NDY = NF1+N
      NPW = NDY+N
  120 JS = JSKF
      CALL LDASUB (Y,YL,T,TEND,N,NY,M,JS,KF,MAXDER,IPRT,H,HMIN,HMAX,
     1RMSEPS,W,W(NSVL),W(NYMAX),W(NER),W(NESV),W(NF1),W(NDY),W(NPW))
C
C     CODE JSKF ON RETURN FROM LDASUB
C
      JSKF = ISIGN(JS*10+IABS(KF),KF)
      RETURN
  130 JSKF = -6
      RETURN
  140 PRINT 1, JSKF
      STOP
C
C
    1 FORMAT ('0IT IS AN ERROR TO ENTER SDESOL WITH JSKF = ',I10//
     1 '    RUN HAS BEEN TERMINATED.')
      END


      SUBROUTINE LDASUB (Y,YL,T,TEND,N,NY,M,JSTART,KFLAG,MAXDR,IPRT,H,
     1HMIN,HMAX,RMSEPS,SAVE,YLSV,YMAX,ER,ESV,F1,DY,PW)
C
C     SUBROUTINE LDASUB IS A MODIFICATION OF SUBROUTINE DFASUB
C     WHICH IS DUE TO R. L. BROWN AND C. W. GEAR.  DFASUB IS DOCUMENTED
C     IN THE REPORT
C        DOCUMENTATION FOR DFASUB--
C        BY R. L. BROWN AND C. W. GEAR
C        REPORT UIUCDCS-R-73-575, JULY 1973
C        UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN
C        URBANA, ILLINOIS  61801
```

```
C      THIS REPORT IS AVAILABLE FROM THE NATIONAL TECHNICAL INFORMATION
C      SERVICE OF THE U. S. DEPARTMENT OF COMMERCE UNDER ACCESSION NUMBER
C      COO-1469-225.
C
C      THE MODIFICATION HERE IS DOCUMENTED IN THE REPORT
C         A PROGRAM FOR THE NUMERICAL SOLUTION OF LARGE SPARSE SYSTEMS OF
C         ALGEBRAIC AND IMPLICITLY DEFINED STIFF DIFFERENTIAL EQUATIONS
C         BY RICHARD FRANKE
C         REPORT NPS53FE76051, MAY 1976
C         NAVAL POSTGRADUATE SCHOOL
C         MONTEREY, CALIFORNIA  93940
C
C      -----------------------------------------------------------------
C
C      THE CALLING SEQUENCE FOR LDASUB IS
C
C      CALL LDASUB(Y,YL,T,TEND,N,NY,M,JSTART,KFLAG,MAXOR,IPRT,H,HMIN,
C      HMAX,RMSEPS,SAVE,YLSV,YMAX,ER,ESV,F1,DY,PW)
C
C      WHERE THE PARAMETERS ARE DEFINED AS FOLLOWS.
C         Y          - ARRAY DIMENSIONED (7,NY).  THIS ARRAY CONTAINS THE
C                      DEPENDENT VARIABLES AND THEIR SCALED DERIVATIVES.
C                      Y(J+1,I) CONTAINS THE J-TH DERIVATIVE OF THE I-TH VAR
C                      IABLE TIMES H**J/J-FACTORIAL, WHERE H IS THE CURRENT
C                      STEPSIZE.  ON FIRST ENTRY THE CALLER SUPPLIES THE
C                      INITIAL VALUES OF EACH VARIABLE IN Y(1,I) AND AN
C                      ESTIMATE OF THE INITIAL VALUES OF THE DERIVATIVES
C                      IN Y(2,I).  ON SUBSEQUENT ENTRIES IT IS ASSUMED THAT
C                      THE ARRAY HAS NOT BEEN CHANGED.  TO INTERPOLATE TO
C                      NON-MESH POINTS, THESE VALUES CAN BE USED AS FOLLOWS.
C                      IF H IS THE CURRENT STEPSIZE AND VALUES AT TIME T+E
C                      NEEDED, LET S = E/H  AND THEN
C
C                                                       NQ
C                           I-TH VARIABLE AT  T+E IS   SUM Y(J+1,I)*S**J
C                                                       J=0
C
C                      THE VALUE OF NQ IS OBTAINED IN THE CALLING PROGRAM
C                      BY  NQ = JSTART.
C
C         YL         - ARRAY OF NL = N - NY VARIABLES WHICH APPEAR LINEARLY.
C                      THE USER SUPPLIES INITIAL VALUES FOR THESE VARIABLES.
C         T          - CURRENT VALUE OF THE INDEPENDENT VARIABLE (TIME)
C         TEND       - END TIME
C         N          - TOTAL NUMBER OF VARIABLES
C         NY         - NUMBER OF DIFFERENTIAL EQUATIONS AND NONLINEAR
C                      VARIABLES.
C         M          - NUMBER OF VARIABLES INCLUDED IN THE ERROR TEST.
C                      THIS NUMBER CAN BE NO GREATER THAN NY.  IF IT IS
C                      GREATER THAN NY, NY VARIABLES ARE USED IN THE ERROR
C                      TEST.
C         JSTART     - INPUT AND OUTPUT INDICATOR.
C                      ON INPUT JSTART HAS THE FOLLOWING MEANINGS.
C                              <0   THIS INDICATES A RE-START FROM A PREVIOUS
C                                   POINT FOLLOWING TERMINATION OF THE RUN OR
C                                   SOLUTION OF ANOTHER PROBLEM DURING THE SAME
C                                   RUN.  PARAMETERS IN THE CALLING SEQUENCE
C                                   MUST HAVE BEEN PRESERVED FROM THE PREVIOUS
C                                   USE, PARTICULARLY THE ARRAYS
C                                        SAVE, YLSV, ESV, AND PW.
C                                   THESE ARRAYS MUST BE SAVED AFTER A CALL
C                                   TO SUBROUTINE LDASAV, WHICH ALSO SAVES
C                                   NECESSARY PARAMETERS INTERNAL TO LDASUB.
C                              =0   INDICATES AN INITIAL CALL TO LDASUB.  THE
C                                   ROUTINE INITIALIZES ITSELF, SCALES THE
C                                   DERIVATIVES IN Y(2,I) AND THEN PERFORMS THE
C                                   INTEGRATION UNTIL T > TEND.
C                              >0   INDICATES THE SOLUTION IS TO BE CONTINUED.
C                                   AFTER THE INITIAL ENTRY IT IS NEITHER
C                                   DESIRABLE NOR NECESSARY TO RE-ENTER WITH
C                                   JSTART = 0, SINCE THIS RE-INITIALIZES
C                                   THE CODE, BEGINNING WITH A FIRST ORDER
C                                   METHOD AGAIN.
C                      ON OUTPUT, JSTART IS SET TO THE VALUE OF NQ, THE
C                      ORDER OF THE FORMULA CURRENTLY BEING USED.
```

```
C        KFLAG    - THE COMPLETION CODE INDICATOR, WITH THE FOLLOWING
C                   MEANINGS
C                         +1   THE INTEGRATION WAS SUCCESSFUL
C                         -1   ERROR TEST FAILURE FOR H > HMIN
C                         -3   CORRECTOR FAILED TO CONVERGE FOR H > HMIN
C                         -4   CORRECTOR FAILED TO CONVERGE FOR FIRST
C                              ORDER METHOD
C                         -5   ERROR RETURN FROM SUBROUTINE NUITSL
C        MAXDER   - MAXIMUM ORDER DERIVATIVE THAT SHOULD BE USED IN THE
C                   METHOD.  IT MUST BE NO GREATER THAN SIX.  IF IT IS
C                   GREATER THAN SIX, THE MAXIMUM ORDER USED WILL BE SIX.
C        IPRT     - INTERNAL PRINT CONTROL INDICATOR
C                         = 0   NO PRINT
C                         > 0   PRINT COUNTERS, STEPSIZE, CURRENT TIME
C                               AND VALUES OF DEPENDENT VARIABLES AT
C                               EACH STEP.
C        H        - CURRENT STEPSIZE.  AN INITIAL VALUE MUST BE SUPPLIED
C                   BUT NEED NOT BE THE ONE WHICH WILL BE USED, SINCE THE
C                   SUBROUTINE WILL CHOOSE A SMALLER ONE IF NECCESSARY TO
C                   KEEP THE ERROR PER STEP SMALLER THAN THE SPECIFIED
C                   VALUE.  IT IS BETTER TO UNDERESTIMATE THE INITIAL
C                   STEPSIZE THAN TO OVERESTIMATE IT.  THE STEPSIZE IS
C                   NORMALLY NOT CHANGED BY THE USER.
C        HMIN     - MINIMUM STEPSIZE ALLOWED
C        HMAX     - MAXIMUM STEPSIZE ALLOWED
C        RMSEPS   - THE ERROR TEST CONSTANT.  THE ROOT-MEAN-SQUARE OF
C                   THE SINGLE STEP ERROR ESTIMATES, ER(I), DIVIDED BY
C                   YMAX(I) = (MAXIMUM TO CURRENT TIME OF Y(I)) MUST BE
C                   LESS THAN RMSEPS.  THE STEPSIZE AND/OR ORDER ARE
C                   VARIED TO ACHIEVE THIS.
C        SAVE     - AN ARRAY OF LENGTH AT LEAST 7*NY
C        YLSV     - AN ARRAY OF LENGTH AT LEAST NL
C        YMAX     - A VECTOR OF LENGTH NY WHICH CONTAINS THE MAXIMUM
C                   OF EACH Y SEEN SO FAR.  ON THE FIRST CALL, THESE WILL
C                   BE INITIALIZED AS YMAX(I) = MAX(1,|Y(1,I)|)
C        ER       - A VECTOR OF LENGTH NY
C        ESV      - A VECTOR OF LENGTH NY
C        F1       - A VECTOR OF LENGTH N = NY + NL
C        DY       - A VECTOR OF LENGTH N = NY + NL
C        PW       - AN ARRAY IN WHICH THE  J  MATRIX COMPUTED
C                   IN SUBROUTINE JACMAT WILL BE STORED.  SIZE WHICH
C                   MUST BE ALLOWED IS DETERMINED BY THE STORAGE TECH-
C                   NIQUE USED FOR IT, BUT NORMALLY WON'T BE MORE THAN
C                   N**2 + 2*N LOCATIONS, THE LATTER 2*N BEING REQUIRED
C                   BY THE LINEAR EQUATION SOLVER.
C
C     -----------------------------------------------------------------
      DIMENSION Y(7,1), YL(1), SAVE(7,1), YMAX(1), ER(1), YLSV(1), F1(1)
     1, PERT(6,3), COF(21), ESV(1), DY(1), PW(1), SAV(1), A(29)
      EQUIVALENCE (A(8),BND), (A(9),BR), (A(10),E), (A(11),EDWN),
     1(A(12),ENQ1), (A(13),ENQ2), (A(14),ENQ3), (A(15),EPS), (A(16),EUP)
     2,(A(17),HNEW), (A(18),PEPSH), (A(19),IDOUB), (A(20),IWEVAL),
     3(A(21),K), (A(22),LCOPYL), (A(23),LCOPYY), (A(24),MAXDER),
     4(A(25),M1), (A(26),NL), (A(27),NQ), (A(28),NS), (A(29),NW)
C
C     -----------------------------------------------------------------
C
C     THE COEFFICIENTS IN THE PERT ARRAY ARE USED FOR ERROR TESTING AND
C     CHANGING STEPSIZE AND NEED TO BE ACCURATE TO ONLY A FEW DIGITS.
C
C     -----------------------------------------------------------------
      DATA PERT/4.,9.,16.,25.,36.,49.,9.,16.,25.,36.,49.,64.,1.,1.,.25,
     12.7889E-2,1.70569E-3,6.83929E-5/
C     -----------------------------------------------------------------
C
C     THE ENTRIES IN THE COF ARRAY ARE THE COEFFICIENTS FOR THE STIFFLY
C     STABLE METHODS USED IN THIS PROGRAM AND ARE TO BE THE MACHINE
C     PRECISION EQUIVALENTS OF THE FOLLOWING CONSTANTS.
C
C     -1
C     -3/2 , -1/2
C     -11/6 , -1 , -1/6
C     -25/12 , -35/24 , -5/12 , -1/24
C     -137/60 , -15/8 , -17/24 , -1/8 , -1/120
C     -147/60, -203/90 , -49/48 , -35/144 , -7/240 , -1/720
```

```
C
C     ------------------------------------------------------------------
      DATA COF/-1.,-1.5,-.5,-1.833333,-1.,-.1666667,-2.083333,-1.458333,
     1-.4166667,-.04166667,-2.283333,-1.875,-.7083333,-.125,-.008333333,
     2-2.45,-2.255556,-1.020833,-.2430556,-.02916667,-.001388889/
      IF (JSTART) 100,110,150
C     ------------------------------------------------------------------
C     IF THIS IS A RESTART ENTRY, RESTORE Y AND YL FROM THE SAVE AND
C     YLSV ARRAYS, WHERE THEY WERE SAVED BY A PREVIOUS CALL TO LDASAV.
C     ------------------------------------------------------------------
  100 CALL COPYZ (Y,SAVE,LCOPYY)
      CALL COPYZ (YL,YLSV,LCOPYL)
      GO TO 150
C     ------------------------------------------------------------------
C     IF THIS IS THE FIRST CALL, INITIALIZE YMAX, SCALE DERIVATIVES, AND
C     INITIALIZE INDICATORS AND SET ORDER TO ONE.
C     FOR DOUBLE PRECISION, SET LCOPYL = 14*NY AND LCOPYL = 2*NL IF
C     SUBROUTINE COPYZ IS IN SINGLE PRECISION.
C     ------------------------------------------------------------------
  110 NL = N-NY
      LCOPYY = 7*NY
      LCOPYL = NL
      M1 = MINO(M,NY)
      EPS = SQRT(FLOAT(M1))*RMSEPS
      MAXDER = MINO(MAXOR,6)
      IF (IPRT.LE.0) GO TO 120
      PRINT 3, N,NL,RMSEPS,TEND,H
      PRINT 4
  120 NS = 0
      NW = 0
C
      DO 130 J=1,NY
      YMAX(J) = AMAX1(1.,ABS(Y(1,J)))
  130 Y(2,J) = Y(2,J)*H
C
      NQ = 1
      RR = 1.
      ASSIGN 190 TO IRET
C     ------------------------------------------------------------------
C     SET COEFFICIENTS FOR THE ORDER CURRENTLY BEING USED.
C     E IS A TEST FOR ERRORS OF THE CURRENT ORDER NQ
C     EUP IS TO TEST FOR INCREASING THE ORDER, EDWN FOR DECREASING THE
C     ORDER.
C     ------------------------------------------------------------------
  140 K = NQ*(NQ-1)/2
      CALL COPYZ (A(2),COF(K+1),NQ)
      K = NQ+1
      IDOUB = NQ
      ENQ1 = .5/NQ
      ENQ2 = .5/K
      ENQ3 = .5/(NQ+2)
      PEPSH = EPS**2
      E = PERT(NQ,1)*PEPSH
      EUP = PERT(NQ,2)*PEPSH
      EDWN = PERT(NQ,3)*PEPSH
      BND = (EPS*ENQ3)**2
      IWEVAL = 1
      GO TO IRET, (190,200,490,570)
  150 IF (H.EQ.HNEW) GO TO 190
C     ------------------------------------------------------------------
C     IF CALLER HAS CHANGED H, RESCALE DERIVATIVES TO REFLECT THAT HNEW
C     WAS USED ON THE LAST CALL.
C     ------------------------------------------------------------------
      R = H/HNEW
      ASSIGN 190 TO IRET
      GO TO 610
C     ------------------------------------------------------------------
C     SET JSTART TO NQ, THE CURRENT ORDER OF THE METHOD, BEFORE EXIT,
C     AND SAVE THE CURRENT STEPSIZE IN HNEW.
C     ------------------------------------------------------------------
  160 JSTART = NQ
      HNEW = H
      RETURN
  170 NS = NS+1
      IF (IPRT.LE.0) GO TO 180
```

```
C     ------------------------------------------------------------------
C     PRINT DATA IF DESIRED BY USER
C     ------------------------------------------------------------------
      PRINT 1, NS,NW,NQ,H,T,(Y(1,I),I=1,NY)
      IF (NL.GT.0) PRINT 2, (YL(I),I=1,NL)
  180 CONTINUE
      IF (KFLAG.LT.0) GO TO 160
      IF (T.GE.TEND) GO TO 160
C     ------------------------------------------------------------------
C     TAKE ANOTHER STEP IF T < TEND
C     ------------------------------------------------------------------
      JSTART = 1
C     ------------------------------------------------------------------
C     SAVE DATA FOR TRIAL WITH A SMALLER TIMESTEP IF THIS STEP FAILS
C     ------------------------------------------------------------------
  190 CALL COPYZ  (SAVE,Y,LCOPYY)
      CALL COPYZ  (YLSV,YL,LCOPYL)
      RACUM = 1.
      KFLAG = 1
      HOLD = H
      NQOLD = NQ
      TOLD = T
  200 T = T+H
      HINV = 1./H
C     ------------------------------------------------------------------
C     COMPUTE PREDICTED VALUES BY EFFECTIVELY MULTIPLYING DERIVATIVE
C     VECTOR BY PASCAL TRIANGLE MATRIX
C     ------------------------------------------------------------------
C
      DO 210 J=2,K
      J3 = K+J-1
C
      DO 210 J1=J,K
      J2 = J3-J1
C
      DO 210 I=1,NY
  210 Y(J2,I) = Y(J2,I)+Y(J2+1,I)
C
C
      DO 220 I=1,NY
  220 ER(I) = 0.
C
C     ------------------------------------------------------------------
C     DO UP TO THREE CORRECTOR ITERATIONS.  CONVERGENCE IS OBTAINED WHEN
C     CHANGES ARE LESS THAN BND WHICH IS DEPENDENT ON THE ERROR TEST
C     CONSTANT.  THE SUM OF CORRECTIONS IS ACCUMULATED IN ER(I).  IT IS
C     EQUAL TO THE K-TH DERIVATIVE OF Y TIMES H**K/(K-FACTORIAL*A(K)),
C     AND THUS IS PROPORTIONAL TO THE ACTUAL ERRORS TO THE LOWEST POWER
C     OF H PRESENT, WHICH IS H**K.
C     ------------------------------------------------------------------
C
      DO 270 L=1,3
      CALL DIFFUN (Y,YL,T,HINV,DY)
      IF (IWEVAL.LT.1) GO TO 230
C     ------------------------------------------------------------------
C     IF THERE HAS BEEN A CHANGE OF ORDER OR THERE HAS BEEN TROUBLE
C     WITH CONVERGENCE, PW IS RE-EVALUATED PRIOR TO STARTING THE
C     CORRECTOR ITERATION.  IWEVAL IS THEN SET TO -1 AS AN INDICATOR
C     THAT IT HAS BEEN DONE.  NEWPW IS SET NONZERO TO INDICATE TO
C     SUBROUTINE NUITSL THAT A NEW PW HAS BEEN PROVIDED.
C     ------------------------------------------------------------------
      CALL JACMAT (Y,YL,T,HINV,A(2),N,NY,EPS,DY,F1.PW)
      KFLAG = 1
      IWEVAL = -1
      NW = NW+1
      NEWPW = 1
  230 CALL NUITSL (PW,DY,F1,N,NY,EPS,YMAX,NEWPW,KRRET)
      IF (KRRET.NE.0) GO TO 600
      IF (NL.LE.0) GO TO 250
C
      DO 240 I=1,NL
  240 YL(I) = YL(I)-F1(I+NY)
C
  250 CONTINUE
      DEL = 0.
```

```
C
      DO 260 I=1,NY
      Y(1,I) = Y(1,I)-F1(I)
      Y(2,I) = Y(2,I)+A(2)*F1(I)
      ER(I) = ER(I)+F1(I)
      DEL = DEL+(F1(I)/AMAX1(YMAX(I),ABS(Y(1,I))))**2
  260 CONTINUE
C
      IF (L.GE.2) BR = AMAX1(.9*BR,DEL/DEL1)
      DEL1 = DEL
      IF (AMIN1(DEL,BR*DEL*2.).LE.BND) GO TO 330
  270 CONTINUE
C
C     -----------------------------------------------------------------
C     THE CORRECTIOR ITERATION FAILED TO CONVERGE IN 3 TRIES.  VARIOUS
C     POSSIBILITIES ARE CHECKED FOR.  IF H IS ALREADY HMIN AND PW HAS
C     ALREADY BEEN RE-EVALUATED, A NO CONVERGENCE EXIT IS TAKEN.
C     OTHERWISE THE MATRIX PW IS RE-EVALUATED AND/OR (IN THAT ORDER) THE
C     STEP IS REDUCED TO TRY AND GET CONVERGENCE.
C     -----------------------------------------------------------------
      T = TOLD
      IF (IWEVAL) 280,300,290
  280 IF (H.LE.HMIN*1.00001) GO TO 310
  290 RACUM = RACUM*.25
  300 CONTINUE
      GO TO 560
  310 KFLAG = -3
C     -----------------------------------------------------------------
C     RESTORE Y AND YL AFTER CONVERGENCE FAILURE
C     -----------------------------------------------------------------
  320 CALL COPYZ (Y,SAVE,LCOPYY)
      CALL COPYZ (YL,YLSV,LCOPYL)
      H = HOLD
      NQ = NQOLD
      GO TO 170
C     -----------------------------------------------------------------
C     THE CORRECTOR CONVERGED, SO NOW THE ERROR TEST IS MADE.
C     -----------------------------------------------------------------
  330 D = 0.
C
      DO 340 I=1,M1
      YM = AMAX1(ABS(Y(1,I)),YMAX(I))
  340 D = D+(ER(I)/YM)**2
C
      IWEVAL = 0
      IF (D.GT.E) GO TO 380
C     -----------------------------------------------------------------
C     THE ERROR TEST IS OKAY, SO THE STEP IS ACCEPTED.  IF IDOUB
C     NOW BECOMES NEGATIVE, A TEST IS MADE TO SEE IF THE STEP SIZE
C     CAN BE INCREASED AT THIS ORDER OR ONE HIGHER OR ONE LOWER.
C     THE CHANGE IS MADE ONLY IF THE STEP CAN BE INCREASED BY AT
C     LEAST 10%.  IDOUB IS SET TO NQ TO PREVENT FURTHER TESTING
C     FOR A WHILE.  IF NO CHANGE IS MADE, IDOUB IS SET TO 9.
C     -----------------------------------------------------------------
      IF (K.LT.3) GO TO 360
C
      DO 350 J=3,K
C
      DO 350 I=1,NY
  350 Y(J,I) = Y(J,I)+A(J)*ER(I)
C
  360 KFLAG = 1
      IDOUB = IDOUB-1
      IF (IDOUB) 410,370,510
  370 CALL COPYZ (ESV,ER,M1)
      GO TO 510
C     -----------------------------------------------------------------
C     THE ERROR TEST FAILED.  IF JSTART = 0, THE DERIVATIVES IN THE
C     SAVE ARRAY ARE UPDATED.  TESTS ARE THEN MADE TO FIX THE STEPSIZE
C     AND PERHAPS REDUCE THE ORDER.  AFTER RESTORING AND SCALING THE
C     Y VARIABLES, THE STEP IS RETRIED.
C     -----------------------------------------------------------------
  380 IF (JSTART.GT.0) GO TO 400
C
      DO 390 I=1,NY
```

```
  390 SAVE(2,I) = Y(2,I)
C
  400 KFLAG = KFLAG-2
      IF (H.LE.HMIN) GO TO 550
      T = TOLD
      IF (KFLAG.LE.-5) GO TO 530
  410 PR2 = (D/E)**ENQ2*1.2
      L = 0
      IF (NQ.LE.1) GO TO 430
      D = 0.
C
      DO 420 J=1,M1
      YM = AMAX1(ABS(Y(1,J)),YMAX(J))
  420 D = D+(Y(K,J)/YM)**2
C
      PR1 = (D/EDWN)**ENQ1*1.3
      IF (PR1.GE.PR2) GO TO 430
      PR2 = PR1
      L = -1
  430 IF (KFLAG.LT.0.OR.NQ.GE.MAXDER) GO TO 450
      D = 0
C
      DO 440 J=1,M1
      YM = AMAX1(ABS(Y(1,J)),YMAX(J))
  440 D = D+((ER(J)-ESV(J))/YM)**2
C
      PR1 = (D/EUP)**ENQ3*1.4
      IF (PR1.GE.PR2) GO TO 450
      PR2 = PR1
      L = 1
  450 R = 1./AMAX1(PR2,1.E-5)
      IF (KFLAG.LT.0.OR.R.GE.1.1) GO TO 460
      IDOUB = 9
      GO TO 510
  460 NEWQ = NQ+L
      K = NEWQ+1
      IF (NEWQ.LE.NQ) GO TO 480
      R1 = A(NEWQ)/FLOAT(NEWQ)
C
      DO 470 J=1,NY
  470 Y(K,J) = ER(J)*R1
C
  480 CONTINUE
C     ------------------------------------------------------------------
C     IF THE STEP WAS OKAY, SCALE THE Y VARIABLES IN ACCORDANCE
C     WITH THE NEW VALUE OF H.  IF KFLAG < 0, HOWEVER, USE THE
C     SAVED VALUES (IN SAVE AND YLSV).  IN EITHER CASE, IF THE ORDER
C     HAS CHANGED IT IS NECESSARY TO FIX CERTAIN PARAMETERS BY CALLING
C     THE PROGRAM SEGMENT AT STATEMENT NUMBER 140.
C     ------------------------------------------------------------------
      IDOUB = NQ
      IF (NEWQ.EQ.NQ) GO TO 490
      NQ = NEWQ
      ASSIGN 490 TO IRET
      GO TO 140
  490 IF (KFLAG.GT.0) GO TO 500
      RACUM = RACUM*R
      GO TO 560
  500 R = AMAX1(AMIN1(HMAX/H,R),HMIN/H)
      H = H*R
      IWEVAL = 1
      ASSIGN 510 TO IRET
      GO TO 610
C
  510 DO 520 I=1,M1
  520 YMAX(I) = AMAX1(ABS(Y(1,I)),YMAX(I))
C
      GO TO 170
C     ------------------------------------------------------------------
C     THE ERROR TEST HAS NOW FAILED THREE TIMES, SO THE DERIVATIVES ARE
C     IN BAD SHAPE.  RETURN TO FIRST ORDER METHOD AND TRY AGAIN.  OF
C     COURSE, IF NQ = 1 ALREADY, THEN THERE IS NO HOPE AND WE EXIT WITH
C     KFLAG = -4.
C     ------------------------------------------------------------------
  530 IF (NQ.EQ.1) GO TO 540
```

```
      NQ = 1
      IDOUB = 1
      ASSIGN 570 TO IRET
      GO TO 140
  540 NQOLD = 1
      KFLAG = -4
      GO TO 320
  550 KFLAG = -1
      GO TO 170
C     ----------------------------------------------------------------------
C     THIS SECTION RESTORES THE SAVED VALUES OF Y AND YL, SCALING THE
C     Y DERIVATIVES AS NECESSARY, AND THEN RETURNS TO THE PREDICTOR LOOP
C     ----------------------------------------------------------------------
  560 H = HOLD*RACUM
      H = AMAX1(HMIN,AMIN1(H,HMAX))
  570 RACUM = H/HOLD
      R1 = 1.
C
      DO 580 J=2,K
      R1 = R1*RACUM
C
      DO 580 I=1,NY
  580 Y(J,I) = SAVE(J,I)*R1
C
C
      DO 590 I=1,NY
  590 Y(1,I) = SAVE(1,I)
C
      CALL COPYZ (YL,YLSV,LCOPYL)
      IWEVAL = 1
      GO TO 200
  600 KFLAG = -5
      GO TO 160
C     ----------------------------------------------------------------------
C     THIS SECTION SCALES THE Y DERIVATIVES BY R**J
C     ---------------------------------------------------------------------
  610 R1 = 1.
C
      DO 620 J=2,K
      R1 = R1*R
C
      DO 620 I=1,NY
  620 Y(J,I) = Y(J,I)*R1
C
      GO TO IRET, (190,510)
C     ----------------------------------------------------------------------
C     THIS SECTION ALLOWS FOR RESTARTS AFTER SOLVING ANOTHER PROBLEM, OR
C     HAVING TERMINATED THE CURRENT COMPUTER RUN.  SUBROUTINE LDASAV
C     SAVES THE NECESSARY VALUES WHICH ARE INTERNAL TO LDASUB.  FOR
C     DOUBLE PRECISION, WITH COPYZ IN SINGLE PRECISION, THE NUMBER OF
C     LOCATIONS TO BE SAVED AND RESTORED, LCOPYS AND LCOPYR. MUST BE
C     SET TO 58.
C     IT IS ASSUMED THAT IN ADDITION TO THE VARIABLES IN THE ARRAY A
C     SAVED BY CALLING LDASAV, THE USER ALSO SAVES THE ARRAYS SAVE,
C     YLSV, YMAX, ESV, AND PW.
C
C     TO RESTART THE USER FIRST CALLS LDARST TO RESTORE THE VALUES SAVED
C     BY LDASAV, THEN RE-ENTERS LDASUB WITH JSTART < 0, AND WITH THE
C     OTHER PARAMETERS THE SAME AS RETURNED FROM THE LAST ENTRY TO
C     LDASUB, PARTICULARLY THOSE ARRAYS MENTIONED ABOVE.
C     ----------------------------------------------------------------------
      ENTRY LDASAV(SAV)
      LCOPYS = 29
      CALL COPYZ (SAV,A,LCOPYS)
      CALL COPYZ (SAVE,Y,LCOPYY)
      CALL COPYZ (YLSV,YL,LCOPYL)
      RETURN
C
      ENTRY LDARST(SAV)
      LCOPYR = 29
      CALL COPYZ (A,SAV,LCOPYR)
      RETURN
C
C     ----------------------------------------------------------------------
C
```

```
C
    1 FORMAT (2I5,I2,1P2E10.2,7E14.6/(32X,7E14.6))
    2 FORMAT (32X,1P7E14.6)
    3 FORMAT ('1   N =',I3,'    NL =',I3,'    RMSEPS =',1PE9.2,'    TEND ='
     1   ,E9.2,'    H =',E9.2//)
    4 FORMAT ('    NS    NW Q     H',8X,'T    ',8X,'Y(1,*) AND YL(*)'//)
      END


      SUBROUTINE COPYZ(S,Y,L)
      DIMENSION S(1),Y(1)
C     -----------------------------------------------------------------
C
C     THIS SUBROUTINE COPIES THE ARRAY Y, OF LENGTH L, INTO THE ARRAY S
C
C     -----------------------------------------------------------------
      IF(L.LE.0)RETURN
      DO 100 J=1,L
  100 S(J) = Y(J)
      RETURN
      END


      SUBROUTINE DERVAL (Y,YL,T,N,NY,W,KERET)
C
C     THIS SUBROUTINE CALCULATES THE INTIAL VALUES OF THE DERIVATIVES
C     IN THE GENERAL CASE.  IT IS WRITTEN SO THAT IT SHOULD WORK IF THE
C     FIRST NY EQUATIONS ALL INVOLVE DERIVATIVES.  IT ATTEMPTS TO SOLVE
C     THE FIRST NY EQUATIONS USING NEWTON'S METHOD, BUT SINCE IT TRIES
C     TO EVALUATE DF/DY' BY CALLING JACMAT IN SUCH A WAY AS TO MAKE THE
C     DF/DY TERM INSIGNIFICANT, IT IS POSSIBLE THAT IT MAY FAIL FOR THAT
C     REASON.  IT MAY FAIL FOR OTHER REASONS, AS WELL.  IF IT DOES FAIL
C     THE USER CAN SUPPLY HIS OWN VERSION OF DERVAL, OR MODIFY THIS
C     ROUTINE IN SUITABLE FASHION.  THIS ROUTINE ASSUMES THAT VALUES OF
C     THE LINEAR VARIABLES HAVE BEEN SUPPLIED PREVIOUSLY.  IF THOSE
C     MUST BE SOLVED FOR SIMULTANEOUSLY WITH THE DERIVATIVES, THE USER
C     MUST SUPPLY HIS OWN VERSION OF DERVAL.
C
C     THE CALLING SEQUENCE FOR THIS SUBROUTINE IS
C
C     CALL DERVAL(Y,YL,T,N,NY,W,KERET)
C
C     WHERE THE PARAMETERS ARE DEFINED AS FOLLOWS
C
C        Y       - SAME AS IN LDASUB AND SDESOL.  Y(1,I) CONTAINS THE
C                  INITIAL VALUES OF THE DEPENDENT VARIABLES.    THE
C                  VALUES OF THE DERIVATIVES ARE RETURNED IN Y(2,I).
C        YL      - SAME AS IN LDASUB AND SDESOL.  THE INITIAL VALUES OF
C                  THE LINEAR VARIABLES MUST BE SUPPLIED TO THIS VERSION
C        T       - INITIAL TIME
C        N       - SAME AS IN LDASUB, TOTAL NUMBER OF VARIABLES
C        NY      - SAME AS IN LDASUB, NUMBER OF DIFFERENTIAL EQUATIONS
C                  AND NONLINEAR VARIABLES
C        W       - SCRATCH ARRAY W FROM THE CALLING SEQUENCE OF SDESOL.
C                  THIS CAN BE USED AS NEEDED IN THIS SUBROUTINE.
C        KERET   - RETURN INDCATOR
C                     =0  NORMAL RETURN
C                     =1  ERROR RETURN
C
C     -----------------------------------------------------------------
      DIMENSION Y(7,1), YL(1), W(1)
C
      DO 100 I=1,NY
      W(2*N+I) = AMAX1(ABS(Y(1,I)),1.)
  100 Y(3,I) = 0.
C
      HINV = 16.**20
      KERET = 0
      EPS2 = NY/1.E8
      EPS = SQRT(EPS2)
C
      DO 140 IT=1,10
C
      DO 110 I=1,NY
  110 Y(2,I) = Y(3,I)/HINV
```

```
C
      CALL DIFFUN (Y,YL,T,HINV,W)
      CALL JACMAT (Y,YL,T,HINV,-1.,NY,NY,EPS,W,W(N+1),W(3*N+1))
      NEWPW = 1
C
      DO 120 I=1,NY
  120 W(I) = W(I)*HINV
C
      CALL NUITSL (W(3*N+1),W,W(N+1),NY,NY,EPS,W(2*N+1),NEWPW,KRET)
      IF (KRET.NE.0) GO TO 170
      ER = 0.
C
      DO 130 I=1,NY
      Y(3,I) = Y(3,I)-W(N+I)
  130 ER = ER+(W(N+I)/AMAX1(ABS(Y(3,I)),1.))**2
C
      IF (ER.LT.EPS2) GO TO 150
  140 CONTINUE
C
      GO TO 170
C
  150 DO 160 I=1,NY
  160 Y(2,I) = Y(3,I)
C
      RETURN
  170 KERET = 1
      RETURN
      END

      SUBROUTINE JACMAT (Y,YL,T,HINV,A2,N,NY,EPS,DY,F1,PW)
C     ------------------------------------------------------------------
C
C     SUBROUTINE JACMAT IS (USUALLY) SUPPLIED BY THE USER.  ITS PURPOSE
C     IS TO EVALUATE THE  J  MATRIX NEEDED WHEN THE CORRECTOR EQUATION
C     IS SOLVED BY NEWTON'S METHOD.  THIS VERSION APPROXIMATES
C     J  BY NUMERICAL DIFFERENCING AND USES FULL STORAGE MODE
C     IN AN NXN MATRIX.
C
C
C     JACMAT CALCULATES THE MATRIX
C
C                  DF       A2  DF
C        J   =     --   -   --  --
C                  DY       H   DY'
C
C     THE CALLING SEQUENCE FOR THIS SUBROUTINE IS
C
C     CALL JACMAT(Y,YL,T,HINV,A2,EPS,N,NY,DY,F1,PW)
C     WHERE THE PARAMETERS ARE DEFINED AS FOLLOWS.
C
C        Y         - SAME AS IN LDASUB AND IN SDESOL.  ON INPUT TO THIS
C                    SUBROUTINE THE ARRAY CONTAINS CURRENT VALUES OF THE
C                    DEPENDENT VARIABLES AND THEIR (SCALED) DERIVATIVES.
C        YL        - SAME AS IN LDASUB AND IN SDESOL.  ON INPUT TO THIS
C                    SUBROUTINE THE ARRAY CONTAINS CURRENT VALUES OF THE
C                    LINEAR VARIABLES.
C        T         - CURRENT TIME
C        HINV      - 1/H , WHERE H IS THE CURRENT STEPSIZE
C        A2        - A(2) FROM LDASUB.
C        N         - SAME AS IN LDASUB, TOTAL NUMBER OF VARIABLES
C        NY        - SAME AS IN LDASUB, NUMBER OF DIFFERENTIAL  EQUATIONS
C                    AND NONLINEAR VARIABLES
C        EPS       - L2 ERROR CONSTANT USED IN LDASUB:
C        DY        - ARRAY OF FUNCTION VALUES AT CURRENT VALUES OF THE
C                    VARIABLES, INPUT TO JACMAT.
C        F1        - SCRATCH ARRAY OF N LOCATIONS WHICH CAN BE USED BY
C                    THIS SUBROUTINE IN ANY WAY NEEDED.
C        PW        - J  MATRIX, OR APPROXIMATION, CALCULATED IN JACMAT AND
C                    RETURNED TO CALLING PROGRAM.  THIS MATRIX IS USED IN
C                    SUBROUTINE NUITSL AND STORAGE MODE MUST AGREE BETWEEN
C                    THE TWO SUBROUTINES.
C
C     ------------------------------------------------------------------
      DIMENSION DY(1), Y(7,1), YL(1), F1(1), PW(1)
```

```
      NL = N-NY
      NN = N*N
C
      DO 100 I=1,NN
  100 PW(I) = 0.
C
C
      DO 120 J=1,NY
      F = Y(1,J)
      E = Y(2,J)
      R = EPS*AMAX1(EPS,ABS(F),ABS(E))
      Y(1,J) = Y(1,J)+R
      Y(2,J) = Y(2,J)-A2*R
      CALL DIFFUN (Y,YL,T,HINV,F1)
C
      DO 110 I=1,N
  110 PW(I+(J-1)*N) = (F1(I)-DY(I))/R
C
      Y(2,J) = E
  120 Y(1,J) = F
C
      IF (NL.EQ.0) GO TO 150
C
      DO 140 J=1,NL
      F = YL(J)
      R = EPS*AMAX1(EPS,ABS(F))
      YL(J) = YL(J)+R
      CALL DIFFUN (Y,YL,T,HINV,F1)
C
      DO 130 I=1,N
  130 PW(I+(J+NY-1)*N) = (F1(I)-DY(I))/R
C
  140 YL(J) = F
C
  150 CONTINUE
      RETURN
      END

      SUBROUTINE NUITSL (PW,DY,F1,N,NY,EPS,YMAX,NEWPW,KRET)
C     ------------------------------------------------------------------
C     THE PURPOSE OF THIS SUBROUTINE IS TO SOLVE A
C     LINEAR SYSTEM OF EQUATIONS FOR THE NEWTON ITERATES WHEN THE
C     CORRECTOR EQUATION IS BEING SOLVED.  UPON ENTRY TO THIS SUBROUTINE
C     THE SYSTEM OF EQUATIONS TO BE SOLVED IS     J W = -F    , WHERE
C        J  IS STORED IN PW UPON ENTRY
C        W  IS RETURNED IN F1
C       -F  IS STORED IN DY UPON ENTRY
C
C     THIS SUBROUTINE IS GENERALLY SUPPLIED BY THE USER, ALTHOUGH THERE
C     ARE SOME STANDARD FORMS AVAILABLE.  FOR EXAMPLE, THIS VERSION
C     ASSUMES THAT PW IS STORED IN FULL STORAGE MODE IN AN NXN MATRIX.
C     IF NEWPW = 1, AN LU DECOMPOSITION IS DONE, NEWPW IS SET TO ZERO
C     AND FORWARD AND BACKWARD SUBSTITUTION FOR THE SOLUTION IS DONE.
C     IF NEWPW = 0, ONLY FORWARD AND BACKWARD SUBSTITUTION FOR THE
C     SOLUTION IS NECESSARY.
C
C     NOTE THAT THIS VERSION OF NUITSL REQUIRES THAT PW HAVE N**2 + 2*N
C     LOCATIONS SINCE 2*N LOCATIONS ARE USED BY THE IMSL LINEAR EQUATION
C     SOLVER.
C
C     NOTE THAT THE PARAMETERS EPS AND YMAX ARE USEFUL IF AN ITERATIVE
C     METHOD IS USED TO SOLVE THE SYSTEM OF EQUATIONS.
C
C     THE CALLING SEQUENCE FOR THIS SUBROUTINE IS
C
C     CALL NUITSL(PW,DY,F1,N,NY,EPS,YMAX,NEWPW,KRET)
C
C     WHERE THE PARAMETERS ARE DEFINED AS FOLLOWS.
C
C        PW        - THE J MATRIX CALCULATED IN SUBROUTINE JACMAT
C        DY        - THE RIGHT HAND SIDE OF THE LINEAR SYSTEM TO BE SOLVED
C        F1        - THE SOLUTION IS RETURNED IN THE ARRAY F1
C        N         - SAME AS IN LDASUB, TOTAL NUMBER OF VARIABLES
C        NY        - SAME AS IN LDASUB, NUMBER OF DIFFERENTIAL EQUATIONS
```

```
C                AND NONLINEAR VARIABLES
C       EPS      - L2 ERROR CONSTANT USED IN LDASUB
C       YMAX     - MAXIMUM VALUES OF Y(1,I) SEEN UP TO THE CURRENT TIME
C       NEWPW    - INDICATES WHETHER A NEW J MATRIX HAS BEEN COMPUTED
C                         =1  INDICATES A NEW J MATRIX HAS BEEN COMPUTED
C                             SINCE THE LAST ENTRY TO NUITSL.  NEWPW
C                             SHOULD BE SET TO ZERO IF SOME PREPROCESSING
C                             SUCH AS LU DECOMPOSITION MUST BE DONE ON A
C                             NEW  J  MATRIX.
C                         =0  INDICATES THE J MATRIX IS THE SAME AS WHEN
C                             NUITSL WAS LAST ENTERED
C       KRET     - RETURN INDICATOR
C                         =0  NORMAL RETURN
C                         =1  ERROR RETURN.  SOLUTION OF EQUATIONS COULD
C                             NOT BE OBTAINED.
C
C     -----------------------------------------------------------------
      DIMENSION PW(1), DY(1), F1(1), YMAX(1)
      NL = N-NY
      IF (NEWPW.EQ.0) GO TO 100
      NEWPW = 0
      NN = N**2+1
      NNN = NN+N
      CALL LUDATF (PW,PW,N,N,0,D1,D2,PW(NN),PW(NNN),F1,IER)
      IF (IER.EQ.0) GO TO 100
      KRET = 1
      RETURN
  100 CALL LUELMF (PW,DY,PW(NN),N,N,F1)
      KRET = 0
      RETURN
      END

      SUBROUTINE DIFFUN(Y,YL,T,HINV,DY)
C     -----------------------------------------------------------------
C     SUBROUTINE DIFFUN IS SUPPLIED BY THE USER.  ITS PURPOSE IS TO
C     EVALUATE THE FUNCTIONS AT CURRENT VALUES OF THE VARIABLES.
C
C     THE CALLING SEQUENCE FOR THIS SUBROUTINE IS
C
C     CALL DIFFUN(Y,YL,T,HINV,DY)
C
C     WHERE THE PARAMETERS ARE DEFINED AS FOLLOWS.
C
C       Y        - SAME AS IN LDASUB AND SDESOL.  ON INPUT TO THIS
C                  SUBROUTINE THE ARRAY CONTAINS CURRENT VALUES OF THE
C                  DEPENDENT VARIABLES AND THEIR (SCALED) DERIVATIVES.
C       YL       - SAME AS IN LDASUB AND SDESOL.  ON INPUT TO THIS
C                  SUBROUTINE THE ARRAY CONTAINS CURRENT VALUES OF THE
C                  LINEAR VARIABLES.
C       T        - CURRENT TIME
C       HINV     - 1/H , WHERE H IS THE CURRENT STEPSIZE
C       DY       - RETURNED ARRAY OF FUNCTION VALUES.
C
C     -----------------------------------------------------------------
      DIMENSION Y(7,1),YL(1),DY(1)
C
C     DEFINE YOUR FUNCTION HERE
C
      RETURN
      END
```

Appendix 2: Examples

Example 1: This example is the problem proposed by Gear [3]. The system of equations is

$$\dot{y}_i - S + (R-y_i)^2 + \sum_{j=1}^{4} b_{ij}\, y_j = 0 \ , \ i = 1, 2, 3, 4$$

$$\text{where} \quad R = \frac{1}{2} \sum_{i=1}^{4} y_i \quad \text{and}$$

$$S = \frac{1}{2} \sum_{i=1}^{4} (R-y_i)^2 \ , \quad \text{and}$$

$$\dot{y}_5 + y_1\, \dot{y}_6 + \dot{y}_1\, y_6 = 0$$

$$2y_6 + y_6^3 - y_1 + V_1 - 1 - e^{-t} = 0$$

$$V_1 - V_2 + y_1\, y_6 = 0$$

$$V_1 + V_2 + 5y_1\, y_2 = 0 \ .$$

In the above $b_{11} = b_{22} = b_{33} = b_{44} = 447.501$

$$b_{12} = -b_{34} = b_{21} = -b_{43} = -\ 452.499$$

$$b_{13} = -b_{24} = b_{31} = -b_{42} = -\ 47.499$$

$$b_{14} = -b_{23} = b_{41} = -b_{32} = -\ 52.501 \ .$$

The initial conditions are

$y_i = 1$, $i=1, 2, 3, 4$.

$y_5 = y_0 = 1$

$V_1 = -\ 2$, $V_2 = -\ 3$.

Note that a different version of DERVAL is necessary since $\frac{\partial F}{\partial \dot{y}}$ is singular.

```
      DIMENSION Y(7,6),YL(2),W(150),GI(16)
      COMMON /DAT/G(16)
      DATA GI/447.501,-452.499,-47.499,-52.501,-452.499,447.501,52.501,
     1 47.499,-47.499,52.501,447.501,452.499,-52.501,47.499,452.499,
     2 447.501/
      DATA N,NY,NL,M,REPS,HMAX,HMIN,H,T,TEND/8,6,2,6,1.E-3,5.E2,1.E-10,
     1 1.E-4,0.,1.E3/
      CALL ERRSET(207,256,-1,1)
      CALL ERRSET(208,256,-1,1)
      DO 8 I=1,16
    8 G(I) = GI(I)
      DO 10 I=1,4
   10 Y(1,I) = -1.
      Y(1,5) = 1.
      Y(1,6) = 1.
      YL(1) = -2.
      YL(2) = -3.
      JSKF = 0
      CALL SDESOL(Y,YL,T,TEND,NY,NL,M,JSKF,6,1,H,HMIN,HMAX,REPS,W)
      PRINT 6,JSKF
    6 FORMAT('0   JSKF =',I4)
      STOP
      END


      SUBROUTINE DIFFUN (Y,YL,T,HINV,DY)
      COMMON /DAT/G(4,4)
      DATA TOLD/-13.459/
      DIMENSION Y(7,1),YL(1),DY(1)
      IF(T.EQ.TOLD)GO TO 10
      TMTERM = EXP(-T)
      TOLD = T
   10 CONTINUE
      R = (Y(1,1) + Y(1,2) + Y(1,3) + Y(1,4))/2.
      S = 0.
      DO 20 I=1,4
   20 S = S + (R - Y(1,I))**2/2.
      DO 30 I=1,4
      DY(I) = HINV*Y(2,I) - S +(R - Y(1,I))**2
      DO 25 J=1,4
   25 DY(I) = DY(I) + G(I,J)*Y(1,J)
   30 CONTINUE
      DY(5) = HINV*(Y(2,5) + Y(1,1)*Y(2,6) + Y(2,1)*Y(1,6))
      DY(6) = 2.*Y(1,6) + Y(1,6)**3 - Y(1,1) + YL(1) - 1.-TMTERM
      DY(7) = YL(1) - YL(2) + Y(1,1)*Y(1,6)
      DY(8) = YL(1) + YL(2) + 5.*Y(1,1) * Y(1,2)
      RETURN
      END


      SUBROUTINE DERVAL(Y,YL,T,N,NY,W,KERET)
      DIMENSION Y(7,1),YL(1),W(1)
      KERET = 0
      DO 50 I=1,NY
   50 Y(2,I) = 0.
      HINV = 1.
      CALL DIFFUN(Y,YL,T,HINV,W)
      DO 100 I=1,NY
  100 Y(2,I) = -W(I)
      RETURN
      END
```

Example 2: This example is a small one, contrived to illustrate the possibility of derivatives entering in a nonlinear fashion. The equations are

$$\dot{y}_1 - 98y_1 + 98y_2 = 0$$

$$(\dot{y}_1)^2 + \dot{y}_2 - 198y_1 + e^{-t}\, y_1 + 199y_2 = 0$$

$$V_1 - y_1 - y_2 = 0$$

The initial conditions are

$$y_1 = y_2 = 1 \ , \ V_1 = 2 \ .$$

Note that we have supplied the explicit expression for the Jacobian. Either JACMAT or a modified version of DERVAL must be supplied as the numerical difference approximation to the Jacobian causes DERVAL to fail.

```
      DIMENSION Y(7,2),YL(1),W(50)
      DATA N,NY,NL,M,REPS,HMAX,HMIN,H,T,TEND/3,2,1,2,1.E-5,25.,1.E-10,
     1 1.E-4,0.,50./
      Y(1,1) = 1.
      Y(1,2) = 1.
      YL(1) = 2.
      JSKF = 0
      CALL SDESOL(Y,YL,T,TEND,NY,NL,M,JSKF,6,1,H,HMIN,HMAX,REPS,W)
      PRINT 6,JSKF
    6 FORMAT('0 JSKF =',I4)
      STOP
      END


      SUBROUTINE DIFFUN (Y,YL,T,HINV,DY)
      DIMENSION Y(7,1),YL(1),DY(1)
      DATA TOLD/-79.03/
      IF(T.EQ.TOLD)GO TO 10
      TMTERM = EXP(-T)
      TOLD = T
   10 CONTINUE
      DY(1) = Y(2,1)*HINV - 98.*Y(1,1) + 99.*Y(1,2)
      DY(2) = (Y(2,1)*HINV)**2 + Y(2,2)*HINV - (198. - TMTERM)*Y(1,1) +
     1 199.*Y(1,2)
      DY(3) = YL(1) - Y(1,1) - Y(1,2)
      RETURN
      END


      SUBROUTINE JACMAT(Y,YL,T,HINV,A2,N,NY,EPS,DY,F1,PW)
      DIMENSION Y(7,1),YL(1),F1(1 ),DY(1),PW(N,1)
      AH = A2*HINV
      DO 100 I=1,N
      DO 100 J=1,N
  100 PW(I,J) = 0.
      PW(1,1) = -AH - 98.
      PW(1,2) = 99.
      PW(2,1) = -2.*AH*Y(2,1)*HINV - 198. + EXP(-T)
      PW(2,2) = -AH + 199.
      IF(NL.LE.0)RETURN
      PW(3,1) = -1.
      PW(3,2) = -1.
      PW(3,3) = 1.
      RETURN
      END
```

Example 3: This is another contrived example, this one to illustrate the use of one type of sparse matrix storage, along with the use of iteration to solve the equations (2.4). The system of equations is

$$\overline{A}\,\dot{y} + \overline{B}\,y = 0\ ,$$

where

$$\overline{A} = \begin{pmatrix} 7 & 0 & 0 & -3 & 0 & -1 \\ 2 & 8 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ -3 & 0 & 0 & 5 & 0 & 0 \\ 0 & -1 & 0 & 0 & 4 & 0 \\ 0 & 0 & 0 & -2 & 0 & 6 \end{pmatrix}$$

$$\overline{B} = \begin{pmatrix} .3 & 0 & 0 & .1 & 0 & -.2 \\ 1 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 10 & 0 & 0 & 20 & 0 & 0 \\ 0 & 5 & 0 & 0 & 6 & 0 \\ 0 & 0 & 0 & 57 & 0 & 100 \end{pmatrix}$$

The initial conditions are

$y_1 = 1000$

$y_2 = 0$

$y_3 = -25$

$y_4 = 10$

$y_5 = 0$

$y_6 = -1000$

The matrix storage scheme used for A, B, and the Jacobian, since it has nonzero elements in the same positions as A and B , is that outlined by Gustavson [4]. Briefly, one stores a pointer array (here called JS) which indicates the initial position of new elements in two other arrays, one of which (here called JN) gives the column number of the element stored in the corresponding position of the coefficient arrays (here called A and B).

Thus, for the above problem the arrays stored are

JS:	1	4	6	7	9	11	13					
JN:	1	4	6	2	1	3	4	1	5	2	6	4
A :	7	-3	-1	8	2	1	5	-3	4	-1	6	-2
B :	.3	.1	-.2	3	1	1	20	10	6	5	100	57

The elements of the $i^{\underline{th}}$ row are stored beginning at location JS(i) of the array A , and in particular A(JS(i) + k) is the element in the $i^{\underline{th}}$ row and JN(JS(i) + k) $^{\underline{th}}$ column of $\bar{A}$, for k = 0, 1, ..., JS(i+1) - JS(i) - 1 . For our purposes it is necessary to access the diagonal element easily, so we have required that the diagonal element be the first element stored for a given row. This means that JN(JS(i)) = i , i=1, ..., n . Note that JS(i) is the number of nonzero elements in rows 1 through i - 1 ,and that JS(n+1) must be defined as the total number of nonzero elements.

Problems similar to the above arise when the finite element method is used to discretize the space domain for time dependent partial differential equations. Simple modifications to the subroutines given below should permit solution of large problems arising in that fashion. We note, however, that it is not convenient to store synmetric matrices in this form unless all nonzero elements are stored. Storage of only the elements of the lower triangular matrix requires one to reference columns of the matrix, which are not readily accessible. Even if the entire matrix is stored, total storage requirements for matrices arising in finite element applications is still considerably less with this scheme than that required by symmetric band storage mode [5].

```
      DIMENSION Y(7,6),W(126),YI(6),AD(12),BD(12),JSD(7),JND(12)
      COMMON /DATA/A(12),B(12),N,JS(7),JN(12)
      INTEGER*2 JS,JN
      DATA T,TEND,H,JSKF / 0.,250.,1.E-5,0 /
      DATA JSD/1,4,6,7,9,11,13/
      DATA JND/1,3,5,2,1,3,4,1,5,2,6,4/
      DATA AD/7.,-3.,-1.,8.,2.,1.,5.,-3.,4.,-1.,6.,-2./
      DATA BD/.3,.1,-.2,3.,1.,1.,20.,10.,5.,5.,100.,57./
      DATA YI/1000.,0.,-25.,10.,0.,-1000./
      N = 6
      NP1 = N + 1
      JE = JSD(NP1) - 1
      DO 100 I=1,N
  100 Y(1,I) = YI(I)
      DO 110 I=1,JE
      A(I) = AD(I)
      B(I) = BD(I)
  110 JN(I) = JND(I)
      DO 120 I=1,NP1
  120 JS(I) = JSD(I)
      PRINT 4,N,(JS(I),I=1,NP1)
      PRINT 5,(JN(I),I=1,JE)
      PRINT 7,(A(I),I=1,JE)
      PRINT 6,(B(I),I=1,JE)
      CALL SDESOL(Y,YL,T,TEND,N,0,N,JSKF,6,1,H,1.E-6,125.,1.E-4,W)
      PRINT 3,JSKF
      STOP
    3 FORMAT('0RETURN FROM SDESOL WITH JSKF =',I4)
    4 FORMAT(' FOR THIS CASE N=',I3//' THE JS ARRAY'//(12I10))
    5 FORMAT(/'0THE JN ARRAY'//(12I10))
    6 FORMAT(/'0THE B ARRAY'//(12F10.2))
    7 FORMAT(/'0THE A ARRAY'//(12F10.2))
      END


      SUBROUTINE DIFFUN (Y,YL,T,HINV,DY)
      COMMON /DATA/A(12),B(12),N,JS(7),JN(12)
      INTEGER*2 JS,JN
      DIMENSION Y(7,1),YL(1),DY(1)
      DO 400 I=1,N
      DY(I) = 0.
      JB = JS(I)
      JE = JS(I+1) - 1
      DO 300 J=JB,JE
      DY(I) = DY(I) + Y(2,JN(J))*A(J)*HINV + B(J)*Y(1,JN(J))
  300 CONTINUE
  400 CONTINUE
      RETURN
      END


      SUBROUTINE NUITSL (PW,DY,F1,N,NY,EPS,YMAX,NEWPW,KRET)
      COMMON /DATA/A(12),B(12),NO,JS(7),JN(12)
      INTEGER*2 JS,JN
      DIMENSION PW(1),DY(1),F1(1),YMAX(1)
      DATA OMEG,OMEGM1 /1.05,.05/
      KRET = 0
      EPSS = EPS**2
      EPSA2 = EPSS*.0001
      NCIT = N
      DO 100 I=1,NY
  100 F1(I) = DY(I)/PW(JS(I))
      DO 300 IT=1,NCIT
      RCH = 0.
      CH = 0.
      DO 200 I=1,NY
      JB = JS(I) + 1
      JE = JS(I+1) - 1
      FN = DY(I)
      IF(JB.GT.JE)GO TO 180
      DO 150 J=JB,JE
  150 FN = FN - PW(J)*F1(JN(J))
  180 FN = FN/PW(JB-1)
      FN = FN*OMEG - F1(I)*OMEGM1
      ACH = F1(I) - FN
```

```
      CH = CH + (ACH/YMAX(I))**2
      RCH = RCH + (ACH/AMAX1(ABS(FN),EPS))**2
  200 F1(I) = FN
      IF(RCH.LT.EPSS) RETURN
      IF(CH.LT.EPSA2) RETURN
  300 CONTINUE
      KRET = 1
      RETURN
      END


      SUBROUTINE JACMAT(Y,YL,T,HINV,A2,N,NY,EPS,DY,F1,PW)
      COMMON /DATA/A(12),B(12),NDUM,JS(7),JN(12)
      INTEGER*2 JS,JN
      DIMENSION Y(7,1),YL(1),F1(1),DY(1),PW(1)
      AH = -A2*HINV
      JE = JS(N+1) - 1
      DO 100 J=1,JE
  100 PW(J) = AH*A(J) + B(J)
      RETURN
      END
```

Example 4: This example arises from a nonlinear reactor dynamics problem where the finite element method is used to discretize the space domain. The resulting system of equations has the form

$$\bar{A}\,\dot{y} - \bar{B}\,y + w(\bar{C}\,y)\,y = 0 \ ,$$

where $\bar{C}$ is a matrix with three subscripts. The $i^{\underline{th}}$ equation can be expressed as

$$\sum_{j=1}^{N} (\bar{a}_{i_j}\,\dot{y}_j - \bar{b}_{i_j}\,y_j) + w \sum_{j=1}^{N} \sum_{k=1}^{N} \bar{c}_{ijk}\,y_j\,y_k = 0 \ .$$

In this example $N = 28$, and there are at most seven nonzero elements per row in $\bar{A}$ and $\bar{B}$. The nonlinear term $y_j\,y_k$ appears only if both $\bar{a}_{ij}$ and $\bar{a}_{ik}$ are nonzero. Therefore a different type of sparse matrix storage is used for this problem.

An array, K, dimensioned (28, 7) is used to store (for each row), the columns subscripts for the nonzero elements. For convenience in accessing the diagonal element, we require that $K(i,1) = i$. We can note this matrix is simply the connectivity matrix for the finite element grid. Then the nonzero elements of $\bar{A}$ and $\bar{B}$, are stored in the corresponding portions of the arrays A and B respectively. If there are in fact less than seven nonzero coefficients in a row, the remaining $K(i,j)$ are set to zero.

The storage for $\bar{C}$ is somewhat more complicated. $\bar{C}$ is symmetric (invariant under any permutation of subscripts). The nonlinear term of the $i^{\underline{th}}$ equation was rewritten as

$$w \sum_{j=1}^{N} \sum_{k=1}^{N} \bar{c}_{ijk}\, y_j\, y_k = w \sum_{j=1}^{N} \sum_{k=j}^{N} \bar{d}_{ijk}\, y_j\, y_k ,$$

where

$$\bar{d}_{ijk} = \begin{cases} 0 & k < j \\ \bar{c}_{ijk} & k = j \\ \bar{c}_{ijk} + C_{ikj} , & k > j . \end{cases}$$

The coefficients $\bar{d}_{ijk}$ are then stored in a (28,28) array C in the order the second and third subscripts are given here.

(K(i,1), K(i,1)), ..., (K(i,1), K(i,7)), (K(i,2), K(i,2)), ..., (K(i,2), K(i,7)), ..., (K(i,7), K(i,7)) .

The equations can then be written in the form

$$\sum_{j=1}^{7} [A_{ij}\, \dot{y}_{K(i,j)} - B_{ij}\, y_{K(i,j)}] + \sum_{j=i}^{7} \sum_{k=j}^{7} C_{im_{jk}}\, y_{K(i,j)}\, y_{K(i,k)} = 0$$

where

$$m_{jk} = k + \frac{(j-1)(14-j)}{2} .$$

Because of the large amount of data for this problem the input arrays K, A, B, and C are simply listed along with the programs for this example.

```
      DIMENSION Y(7,28),WS(560)
      COMMON /DATA/A(28,7),B(28,7),C(28,28),N,NNZ,K(28,7)
      INTEGER*2 K
      DATA TEND,HMIN,HMAX,EPS,ZOMEGA/.1,1.E-12,.1,.01,650.903E-14/
      DATA NN,NY,NL/28,28,0/
      NNZ = 7
      CALL ERRSET(207,256,-1,1)
      CALL ERRSET(208,256,-1,1)
      N = NN
      M = N
      NEND = NNZ*(NNZ + 1)/2
      PRINT 10
      DO 110 I=1,NN
      READ 1,(K(I,J) ,J=1,NNZ)
  110 PRINT 11,(K(I,J),J=1,NNZ)
      PRINT 12
      DO 120 I=1,NN
      READ 2,(A(I,J),J=1,NNZ)
      Y(1,I) = 0.
  120 PRINT 15,(A(I,J),J=1,NNZ)
      Y(1,1) = 1.E16
      PRINT 13
      DO 130 I=1,NN
      READ 2,(B(I,J),J=1,NNZ)
  130 PRINT 15,(B(I,J),J=1,NNZ)
      PRINT 14
      DO 140 I=1,NN
      READ 2,(C(I,J),J=1,NEND)
      DO 135 J=1,NEND
  135 C(I,J) =  C(I,J) *ZOMEGA
  140 PRINT 15,(C(I,J),J=1,NEND)
      JSKF = 0
      T = 0.
      H = HMIN*1000.
      CALL SDESOL(Y,YL,T,TEND,NY,NL,M,JSKF,6,1,H,HMIN,HMAX,   EPS,WS)
      PRINT 3,JSKF
      STOP
    1 FORMAT(16I5)
    2 FORMAT(7E11.4)
    3 FORMAT('0 JSKF = ', I3)
   10 FORMAT('1K ARRAY'/)
   11 FORMAT(8X,14I8)
   12 FORMAT(///'0A(I,J)'/)
   13 FORMAT(///'0B(I,J)'/)
   14 FORMAT(///'0C(I,J)'/)
   15 FORMAT(8X,1P7E16.6)
      END

      SUBROUTINE DIFFUN(Y,YL,T,HINV,DY)
      COMMON /DATA/A(28,7),B(28,7),C(28,28),N,NNZ,K(28,7)
      INTEGER*2 K
      DIMENSION Y(7,1),YL(1),DY(1)
      DO 400 I=1,N
      DY(I) = 0.
      DO 300 J=1,NNZ
      IF(K(I,J).LE.0)GO TO 310
      DY(I) = DY(I) + Y(2,K(I,J))*A(I,J)*HINV - B(I,J)*Y(1,K(I,J)) +
     1 C(I,J)*Y(1,K(I,J))*Y(1,K(I,1))
  300 CONTINUE
  310 L = NNZ
      DO 360 J1=2,NNZ
      IF(K(I,J1).LE.0)GO TO 400
      DO 350 J2=J1,NNZ
      L = L + 1
      IF(K(I,J2).LE.0)GO TO 350
      DY(I) = DY(I) + C(I,L)*Y(1,K(I,J1))*Y(1,K(I,J2))
  350 CONTINUE
  360 CONTINUE
  400 CONTINUE
      RETURN
      END

      SUBROUTINE JACMAT(Y,YL,T,HINV,A2,N,NY,EPS,DY,F1,PW)
```

```
      COMMON /DATA/A(28,7),B(28,7),C(28,28),ND,NNZ,K(28,7)
      INTEGER*2 K,P
      DIMENSION Y(7,1),YL(1),F1(1 ),DY(1),PW(NY,1)
      DIMENSION P(7,7)
      DATA P/49*0/,INITP/0/
      IF(INITP.EQ.NNZ)GO TO 99
      INITP = NNZ
      DO 98 L=1,NNZ
      DO 98 M = L,NNZ
   98 P(L,M) = M + (L - 1)*(2*NNZ - L)/2
   99 CONTINUE
      AH =-A2*HINV
      DO 300 I=1,NY
      DO 300 J=1,NNZ
      PW(I,J) = AH*A(I,J) - B(I,J)
       DO 100 L=1,J
      IF(K(I,L).LE.0)GO TO 295
  100 PW(I,J) = PW(I,J) + C(I,P(L,J))*Y(1,K(I,L))
      DO 200 M=J,NNZ
      IF(K(I,M).LE.0)GO TO 295
  200 PW(I,J) = PW(I,J) + C(I,P(J,M))*Y(1,K(I,M))
  295 CONTINUE
  300 CONTINUE
      RETURN
      END


      SUBROUTINE NUITSL(PW,DY,F1,N,NY,EPS,YMAX,NEWPW,KRET)
      DIMENSION PW(NY,1),DY(1),F1(1),YMAX(1)
      COMMON /DATA/A(28,7),B(28,7),C(28,28),ND,NNZ,K(28,7)
      DATA  SPD,SPDM1/1.05,.05/
      INTEGER*2 K
      KRET = 0
      EPSS = EPS**2
      EPSA2 = EPSS*.0001
      NOIT = N
  280 DO 281 I=1,NY
  281 F1(I) = DY(I)/PW(I,1)
      DO 287 IT=1,NOIT
      RCH = 0.
      CH = 0.
      DO 285 I=1,NY
      FN = DY(I)
      DO 284 J=2,NNZ
      IF(K(I,J).LE.0.OR.K(I,J).GT.NY)GO TO 284
      FN = FN - PW(I,J)*F1(K(I,J))
  284 CONTINUE
      FN = FN/PW(I,1)
      FN = FN*SPD - SPDM1*F1(I)
      ACH = F1(I) - FN
      CH = CH + (ACH/YMAX(I))**2
      RCH = RCH + (ACH/AMAX1(ABS(FN),EPS))**2
  285 F1(I) = FN
      IF(RCH.LT.EPSS)GO TO 288
      IF(CH.LE.EPSA2)GO TO 288
  287 CONTINUE
      KRET = 3
  288 CONTINUE
      RETURN
      END
```

INPUT DATA FOR EXAMPLE 4

```
   1       6       2
   2       1       6       7       8       3
   3       2       8       4
   4       3       8       9      10       5
   5       4      10      21
   6      11       7       2       1
   7       2       6      11      12      13       8
   8       3       2       7      13       9       4
   9       4       8      13      14      15      10
  10       5       4       9      15      22      21
  11       6      16      12       7
  12       7      11      16      17      18      13
  13       8       7      12      18      14       9
  14       9      13      18      19      20      15
  15      10       9      14      20      23      22
  16      11      28      17      12
  17      12      16      28      27      18
  18      13      12      17      27      26      19      14
  19      14      18      26      25      20
  20      15      14      19      25      24      23
  21       5      10      22
  22      21      10      15      23
  23      22      15      20      24
  24      23      20      25
  25      20      19      26      24
  26      19      18      27
  27      18      17      28      26
  28      16      27      17
 8.3776E 02 8.3776E 02 4.1888E 02 0.0        0.0        0.0        0.0
 5.0265E 03 4.1888E 02 2.0944E 03 2.5133E 03 2.0944E 03 4.1888E 02 0.0
 1.6755E 03 4.1888E 02 1.6755E 03 4.1888E 02 0.0        0.0        0.0
 5.0265E 03 4.1888E 02 2.0944E 03 2.5133E 03 2.0944E 03 4.1888E 02 0.0
 1.4661E 03 4.1888E 02 1.4661E 03 3.1416E 02 0.0        0.0        0.0
 1.0891E 04 2.9322E 03 4.1888E 03 2.0944E 03 8.3776E 02 0.0        0.0
 2.8484E 04 2.5133E 03 4.1888E 03 6.2832E 03 6.7021E 03 6.2832E 03 4.1888E 03
 2.1782E 04 1.6755E 03 2.0944E 03 4.1888E 03 5.8643E 03 4.1888E 03 2.0944E 03
 2.8484E 04 2.5133E 03 4.1888E 03 6.2832E 03 6.7021E 03 6.2832E 03 4.1888E 03
 1.9059E 04 1.4661E 03 2.0944E 03 4.1888E 03 5.1313E 03 3.1416E 03 1.5708E 03
 2.3457E 04 2.9322E 03 5.0265E 03 8.3776E 03 6.2832E 03 0.0        0.0
 5.3616E 04 6.7021E 03 8.3776E 03 1.0472E 04 1.0891E 04 1.0472E 04 8.3776E 03
 4.6914E 04 5.8643E 03 6.2832E 03 8.3776E 03 1.0053E 04 8.3776E 03 6.2832E 03
 5.3616E 04 6.7021E 03 8.3776E 03 1.0472E 04 1.0891E 04 1.0472E 04 8.3776E 03
 4.4663E 04 5.1313E 03 6.2832E 03 8.3776E 03 8.9535E 03 8.4823E 03 6.2046E 03
 3.2515E 04 5.0265E 03 5.1836E 03 1.0812E 04 1.0472E 04 0.0        0.0
 5.6208E 04 1.0891E 04 1.0812E 04 1.1958E 04 1.1958E 04 1.0812E 04 0.0
 8.0582E 04 1.0053E 04 1.0472E 04 1.0812E 04 1.3312E 04 1.3312E 04 2.1284E 04
 5.6208E 04 1.0891E 04 1.0812E 04 1.1958E 04 1.1958E 04 1.0812E 04 0.0
 6.0750E 04 8.9535E 03 1.0472E 04 1.0812E 04 9.2481E 03 1.0348E 04 9.8764E 03
 3.7699E 03 3.1416E 02 1.5708E 03 1.8850E 03
 2.9531E 04 1.8850E 03 3.1416E 03 6.2046E 03 8.7179E 03
 6.1889E 04 8.7179E 03 8.4823E 03 9.8764E 03 1.2468E 04
 6.3303E 04 1.2468E 04 1.0348E 04 8.8357E 03
 5.7962E 04 9.2481E 03 1.1958E 04 1.4726E 04 898657E 03
 9.4719E 04 1.1958E 04 1.3312E 04 1.7671E 04
 9.4719E 04 1.3312E 04 1.1958E 04 1.4726E 04 1.7671E 04
 5.3014E 04 5.1836E 03 1.4726E 04 1.1958E 04
-1.5816E 09 1.1720E 09 1.0449E 09 0.0        0.0        0.0        0.0
-3.9822E 09 1.0449E 09 6.3535E 08 4.4338E 09 3.3535E 08 1.0449E 09 0.0
-3.1631E 09 1.0449E 09 2.3440E 09 1.0449E 09 0.0        0.0        0.0
-3.9822E 09 1.0449E 09 6.3535E 08 4.4338E 09 6.3535E 08 1.0449E 09 0.0
-4.3361E 09 1.0449E 09 1.8355E 09 1.4879E 09 0.0        0.0        0.0
-6.7925E 09 4.5609E 09 6.7778E 09 6.3535E 08 1.1720E 09 0.0        0.0
-1.5223E 10 4.4338E 09 6.7778E 09 1.9061E 09 1.1212E 10 1.9061E 09 6.7778E 09
-1.3585E 10 2.3440E 09 6.3535E 08 6.7778E 09 9.1218E 09 6.7778E 09 6.3535E 08
-1.5223E 10 4.4338E 09 6.7778E 09 1.9061E 09 1.1212E 10 1.9061E 09 6.7778E 09
-2.4104E 10 1.8355E 09 6.3535E 08 6.7778E 09 7.3355E 09 8.4446E 09-6.0318E 08
-1.3995E 10 4.5609E 09 7.9498E 09 1.3556E 10 1.9061E 09 0.0        0.0
-2.9627E 10 1.1212E 10 1.3556E 10 3.1768E 09 1.7989E 10 3.1768E 09 1.3556E 10
-2.7989E 10 9.1218E 09 1.9061E 09 1.3556E 10 1.5900E 10 1.3556E 10 1.9061E 09
-2.9627E 10 1.1212E 10 1.3556E 10 3.1768E 09 1.7989E 10 3.1768E 09 1.3556E 10
-4.8828E 10 7.3355E 09 1.9061E 09 1.3556E 10 1.0212E 10 1.1621E 10 2.0408E 09
-3.5210E 10 7.9498E 09 1.3692E 10 1.6043E 10 3.1768E 09 0.0        0.0
-8.2408E 10 1.7989E 10 1.6043E 10-3.6945E 08 2.4060E 10 1.3103E 10 0.0
```

-7.2321E 10	1.5900E 10	3.1768E 09	1.3103E 10	1.1476E 10	1.1476E 10	1.6280E 10
-8.2408E 10	1.7989E 10	1.3103E 10	2.4060E 10	-3.6945E 08	1.6043E 10	0.0
-8.7119E 10	1.0212E 10	3.1768E 09	1.6043E 10	2.4798E 10	3.4658E 09	1.3398E 10
-8.2837E 09	1.4879E 09	-6.0318E 08	2.8953E 09			
-4.7095E 10	2.8953E 09	8.4446E 09	2.0408E 09	6.4005E 08		
-1.0136E 11	6.4005E 08	1.1621E 10	1.3398E 10	4.6822E 09		
-1.3280E 11	4.6822E 09	3.4658E 09	2.3750E 10			
-1.3346E 11	2.4798E 10	-3.6945E 08	7.2533E 09	2.3750E 10		
-1.5263E 11	2.4060E 10	1.1476E 10	-3.5688E 09			
-1.4877E 11	1.1476E 10	2.4060E 10	3.3929E 09	-3.5688E 09		
-7.4644E 10	1.3692E 10	3.3929E 09	-3.6945E 08			
4.1888E 02	5.5850E 02	2.7925E 02	0.0	0.0	0.0	0.0
4.1888E 02	2.7925E 02	0.0	0.0	0.0	0.0	1.3963E 02
0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0
2.5133E 03	0.0	8.3776E 02	0.0	5.5850E 02	0.0	0.0
1.3963E 02	2.7925E 02	0.0	0.0	0.0	2.7925E 02	9.7738E 02
0.0	0.0	0.0	5.5850E 02	1.1170E 03	5.5850E 02	0.0
2.2340E 03	9.7738E 02	0.0	8.3776E 02	1.3963E 02	5.5850E 02	0.0
8.3776E 02	2.7925E 02	1.1170E 03	2.7925E 02	0.0	0.0	0.0
1.3963E 02	2.7925E 02	0.0	0.0	0.0	0.0	8.3776E 02
2.7925E 02	0.0	0.0	0.0	1.3963E 02	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0
2.5133E 03	0.0	8.3776E 02	0.0	5.5850E 02	0.0	0.0
1.3963E 02	2.7925E 02	0.0	0.0	0.0	2.7925E 02	9.7738E 02
0.0	0.0	0.0	5.5850E 02	1.1170E 03	5.5850E 02	0.0
2.2340E 03	9.7738E 02	0.0	8.3776E 02	1.3963E 02	5.5850E 02	0.0
4.1888E 02	2.7925E 02	5.5850E 02	0.0	0.0	0.0	0.0
1.3963E 02	2.7925E 02	0.0	0.0	0.0	0.0	4.1888E 02
0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0
6.7021E 03	2.2340E 03	1.9548E 03	8.3776E 02	0.0	0.0	0.0
1.2566E 03	1.1170E 03	0.0	0.0	0.0	0.0	1.6755E 03
5.5850E 02	0.0	0.0	1.3963E 03	6.9813E 02	0.0	0.0
1.1170E 03	2.7925E 02	0.0	1.1170E 03	0.0	0.0	0.0
1.6755E 04	2.2340E 03	1.3963E 03	2.5133E 03	0.0	2.2340E 03	1.3963E 03
8.3776E 02	5.5850E 02	0.0	0.0	0.0	0.0	1.6755E 03
1.1170E 03	0.0	0.0	1.9548E 03	2.6529E 03	0.0	0.0
2.2340E 03	2.7925E 03	1.3963E 03	6.4228E 03	2.6529E 03	2.5133E 03	5.3058E 03
1.3404E 04	0.0	1.1170E 03	1.9548E 03	4.4680E 03	1.9548E 03	8.3776E 02
5.5850E 02	2.7925E 02	0.0	0.0	0.0	1.9548E 03	6.9813E 02
0.0	0.0	0.0	8.3776E 02	1.6755E 03	1.1170E 03	0.0
1.9548E 03	2.5133E 03	1.1170E 03	0.0	1.6755E 03	1.9548E 03	1.8151E 03
1.6755E 04	2.2340E 03	1.3963E 03	2.5133E 03	0.0	2.2340E 03	1.3963E 03
8.3776E 02	5.5850E 02	0.0	0.0	0.0	0.0	1.6755E 03
1.1170E 03	0.0	0.0	1.9548E 03	2.6529E 03	0.0	0.0
2.2340E 03	2.7925E 03	1.3963E 03	6.4228E 03	2.6529E 03	2.5133E 03	5.3058E 03
6.7021E 03	0.0	1.1170E 03	1.9548E 03	2.2340E 03	0.0	0.0
2.7925E 02	2.7925E 02	0.0	0.0	0.0	8.3776E 02	6.9813E 02
0.0	0.0	0.0	8.3776E 02	1.6755E 03	1.1170E 03	0.0
1.9548E 03	1.2566E 03	0.0	0.0	0.0	0.0	0.0
1.4242E 04	0.0	3.9095E 03	3.6303E 03	2.5133E 03	0.0	0.0
1.1170E 03	0.0	0.0	1.1170E 03	0.0	2.5133E 03	2.0944E 03
1.9548E 03	0.0	0.0	0.0	3.3510E 03	1.3963E 03	0.0
3.0718E 03	2.3736E 03	0.0	2.7925E 03	0.0	0.0	0.0
3.1835E 04	5.5850E 03	3.0718E 03	4.1888E 03	0.0	3.9095E 03	3.0718E 03
2.5133E 03	1.3963E 03	0.0	0.0	0.0	0.0	3.3510E 03
1.9548E 03	0.0	0.0	3.6303E 03	4.3284E 03	0.0	0.0
3.9095E 03	4.4680E 03	2.2340E 03	1.0612E 04	4.3284E 03	4.1888E 03	1.0332E 04
2.8484E 04	0.0	2.7925E 03	3.6303E 03	7.8191E 03	3.6303E 03	2.5133E 03
2.2340E 03	1.1170E 03	0.0	0.0	0.0	6.1436E 03	2.3736E 03
0.0	0.0	0.0	2.5133E 03	3.3510E 03	1.9548E 03	0.0
4.4680E 03	4.1888E 03	1.9548E 03	0.0	3.3510E 03	4.4680E 03	5.1662E 03
3.1835E 04	5.5850E 03	3.0718E 03	4.1888E 03	0.0	3.9095E 03	3.0718E 03
2.5133E 03	1.3963E 03	0.0	0.0	0.0	0.0	3.3510E 03
1.9548E 03	0.0	0.0	3.6303E 03	4.3284E 03	0.0	0.0
3.9095E 03	4.4680E 03	2.2340E 03	1.0612E 04	4.3284E 03	4.1888E 03	1.0332E 04
1.4242E 04	0.0	2.7925E 03	3.6303E 03	3.9095E 03	0.0	0.0
1.1170E 03	1.1170E 03	0.0	0.0	0.0	2.5133E 03	2.3736E 03
0.0	0.0	0.0	2.5133E 03	3.3510E 03	1.9548E 03	0.0
4.4680E 03	2.0944E 03	0.0	0.0	0.0	0.0	0.0
1.3823E 04	0.0	0.0	0.0	4.1888E 03	0.0	0.0
1.9548E 03	0.0	0.0	1.9548E 03	0.0	4.1888E 03	0.0
0.0	0.0	0.0	0.0	2.3736E 03	2.2340E 03	0.0
4.7473E 03	4.0492E 03	0.0	4.4680E 03	0.0	0.0	0.0

1.4242E 04	8.9361E 03	4.7473E 03	0.0	0.0	4.7473E 03	0.0
4.1888E 03	2.2340E 03	0.0	0.0	0.0	0.0	2.3736E 03
0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	2.3736E 03	2.2340E 03	0.0
2.7646E 04	0.0	4.4680E 03	0.0	0.0	0.0	4.1888E 03
3.9095E 03	1.9548E 03	0.0	0.0	0.0	1.0332E 04	4.0492E 03
0.0	0.0	0.0	4.1888E 03	2.3736E 03	0.0	0.0
6.9813E 03	0.0	0.0	0.0	0.0	0.0	1.7872E 04
1.4242E 04	8.9361E 03	4.7473E 03	0.0	0.0	4.7473E 03	0.0
4.1888E 03	2.2340E 03	0.0	0.0	0.0	0.0	2.3736E 03
0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	2.3736E 03	2.2340E 03	0.0
1.3823E 04	0.0	4.4680E 03	0.0	0.0	0.0	0.0
1.9548E 03	1.9548E 03	0.0	0.0	0.0	4.1888E 03	4.0492E 03
0.0	0.0	0.0	4.1888E 03	2.3736E 03	0.0	0.0
6.9813E 03	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0

References

1. R. L. Brown and C. W. Gear, "Documentation for DFASUB - a program for the solution of simultaneous implicit differential and nonlinear equations," Report no. UIUCDCS-R-73-575, University of Illinois at Urbana - Champaign, Urbana, Illinois, July 1973.

2. C. W. Gear Numerical Initial Value Problems in Ordinary Differential Equations, Prentice-Hall, Englewood Cliffs, NJ, 1971.

3. C. W. Gear, "Simultaneous Numerical Solution of Differential-Algebraic Systems," IEEE Trans. on Circuit Theory, CT-18(1971)89-95.

4. F. G. Gustavson, "Some Basic Techniques for Solving Sparse Systems of Linear Equations," pp 41-52 in Sparse Matrices and Their Applications, D. J. Rose and R. A. Willoughby, eds., Plenum Press, New York - London, 1972.

5. D. Salinas, D. H. Nguyen, R. Olsen, and R. Franke "An Optimal Compact Storage Scheme for Nonlinear Reactor Problems by FEM," Manuscript (to be submitted).

Distribution List

	No. of copies
Defense Documentation Center Cameron Station Alexandria, VA 22314	12
Library Naval Postgraduate School Monterey, CA 93940	2
Dean of Research Naval Postgraduate School Monterey, CA 93940	2
Naval Postgraduate School Department of Mathematics	
Ladis D. Kovach, Chairman	1
Professor C. Comstock	1
Professor F. Faulkner	1
Professor R. Franke	10
Naval Postgraduate School Department of Mechanical Engineering	
Professor D. Salinas	1
Professor D. Nguyen	1
Professor R. Newton	1
Professor G. Cantin	1
Naval Postgraduate School Department of Aeronautics	
Professor D. Collins	1
Professor R. Ball	1
Naval Postgraduate School Computer Center Monterey, CA 93940	
Professor D. Williams	1
Mr. Roger Hilleary	1
Dr. Richard Lau Office of Naval Research Pasadena, CA 91100	1
Chief of Naval Research ATTN: Mathematics Program Arlington, VA 22217	2

Mr. W. J. Dejka, Code 4000 Naval Electronics Laboratory Center San Diego, CA 92152	1
Argonne National Laboratory Argonne, IL 60439 ATTN: Mr. Gary Leaf, AMD Mr. Tilak Chawla, RAS	 1 1
Professor C. W. Gear University of Illinois Urbana, IL 61801	1
Dr. A. C. Hindmarsh Lawrence Livermore Laboratory Livermore, CA 94550	1
Sandia Laboratories Albuquerque, NM 87115 ATTN: D. A. Dahlgren L. F. Shampine	 1 1
Mr. R. E. Huddleston Sandia Laboratories Livermore, CA 94550	1
Air Force Weapons Laboratory Kirtland AFB Albuquerque, NM 87115 ATTN: C. M. Walters CAPT. C. W. Stein	 1
Mr. R. D. Birkhoff Oak Ridge National Laboratory Union Carbide Corporation P. O. Box X Oak Ridge, TN 37830	1
Mr. R. M. Sternheimer Brookhaven National Laboratory Upton, NY 11973	1
Mr. B. F. Maskewitz RSIC, Neutron Physics Division Oak Ridge National Laboratory Oak Ridge, TN 37831	1
Mr. J. L. Black U. S. Naval Research Laboratory Code 770 Washington, DC 20390	1

Los Alamos Scientific Laboratory P. O. Box 1663 Los Alamos, NM 87544 ATTN: S. Evans, T-6	1
C. Young, J-14	1
Professor R. E. Barnhill Department of Mathematics University of Utah Salt Lake City, Utah 84112	1
LT Donald Hinsman Fleet Numerical Weather Central Monterey, CA 93940	1

U173475

DUDLEY KNOX LIBRARY - RESEARCH REPORTS
5 6853 01071186 4

NPS